ПРОФЕССИОНАЛЬНО

Эндрю Стелльман
Дженнифер Грин

ИДЕАЛЬНАЯ КОМАНДА

ВДОХНОВЛЯЮЩИЕ И ПРЕДОСТЕРЕГАЮЩИЕ РАССКАЗЫ ВЕТЕРАНОВ ТИМЛИДИНГА

СИМВОЛ®

По договору между издательством «Символ-Плюс» и Интернет-магазином «Books.Ru – Книги России» единственный легальный способ получения данного файла с книгой ISBN 978-5-93286-171-4, название «Идеальные команды. Вдохновляющие и предостерегающие рассказы ветеранов тимлидинга» – покупка в Интернет-магазине «Books.Ru – Книги России». Если Вы получили данный файл каким-либо другим образом, Вы нарушили международное законодательство и законодательство Российской Федерации об охране авторского права. Вам необходимо удалить данный файл, атакже сообщить издательству «Символ-Плюс» (piracy@symbol.ru), где именно Вы получили данный файл.

Beautiful Teams

Inspiring and Cautionary Tales from Veteran Team Leaders

Andrew Stellman & Jennifer Greene

O'REILLY®

ПРОФЕССИОНАЛЬНО

ИДЕАЛЬНЫЕ КОМАНДЫ

Вдохновляющие и предостерегающие рассказы ветеранов тимлидинга

Эндрю Стеллман и Дженнифер Грин

СИМВОЛ®

Санкт-Петербург — Москва
2010

Серия «Профессионально»

Эндрю Стеллман, Дженнифер Грин

Идеальные команды

Вдохновляющие и предостерегающие рассказы ветеранов тимлидинга

Перевод А. Коробейникова

Главный редактор	*А. Галунов*
Зав. редакцией	*Н. Макарова*
Выпускающий редактор	*П. Щеголев*
Редактор	*Т. Темкина*
Корректор	*С. Минин*
Верстка	*К. Чубаров*

Стеллман Э., Грин Дж.
Идеальные команды. Вдохновляющие и предостерегающие рассказы ветеранов тимлидинга. – Пер. с англ. – СПб: Символ-Плюс, 2010. – 608 с., ил.

ISBN 978-5-93286-171-4

Как создать эффективную команду? Что необходимо сделать, чтобы организовать работу и создать доброжелательную атмосферу в коллективе? Как тимлидеру держать процесс под контролем?

Книга «Идеальные команды» полно и беспристрастно отвечает на эти и другие вопросы. В ее написании приняли участие мастера высочайшего класса, опытные руководители ИТ-проектов, познавшие как успех, так и неудачи. Яркие индивидуальности делятся с читателями опытом, который не смог бы самостоятельно накопить один человек.

Авторы не навязывают читателю свое мнение, не предлагают застывшие догмы, а помогают определить общие принципы, которые позволяют сплачивать команды, делать их сильнее и эффективнее.

ISBN 978-5-93286-171-4
ISBN 978-0-596-51802-8 (англ)

© Издательство Символ-Плюс, 2010
Authorized translation of the English edition © 2009 O'Reilly Media Inc. This translation is published and sold by permission of O'Reilly Media Inc., the owner of all rights to publish and sell the same.

Все права на данное издание защищены Законодательством РФ, включая право на полное или частичное воспроизведение в любой форме. Все товарные знаки или зарегистрированные товарные знаки, упоминаемые в настоящем издании, являются собственностью соответствующих фирм.

Издательство «Символ-Плюс». 199034, Санкт-Петербург, 16 линия, 7,
тел. (812) 324-5353, www.symbol.ru. Лицензия ЛП N 000054 от 25.12.98.

Подписано в печать 31.05.2010. Формат 70×90 1/16. Печать офсетная.
Объем 38 печ. л. Тираж 1500 экз. Заказ №
Отпечатано с готовых диапозитивов в ГУП «Типография «Наука»
199034, Санкт-Петербург, 9 линия, 12.

Эта книга посвящается Тревору Филду и другим сотрудникам PlayPumps International, посвятившим себя нелегкой работе – обеспечивать чистой питьевой водой жителей Африки к югу от Сахары. Часть доходов от продажи этой книги будет передана на их благородное дело. Больше о них и об их труде вы узнаете, посетив сайт http://www.playpumps.org.

Оглавление

Отзывы 9
Почему «идеальные команды»? 11
Предисловие 19

Глава 1. Лидерство
Интервью с Тимом О'Рейли 27

I. Люди 41

Глава 2. Почему побеждают ужасные команды
Скотт Беркун 45

Глава 3. Создание видеоигр
Интервью с Марком Хили 53

Глава 4. Построение идеальной команды
Билл Дипьер 67

Глава 5. Что мотивирует разработчиков
Интервью с Энди Лестером 83

Глава 6. Как воодушевлять людей
Интервью с Кеоки Андрусом 95

Глава 7. Музыкальную индустрию – в XXI век!
Том Тарка 113

Глава 8. Inner Source – ПО с исходным кодом, открытым для внутреннего использования
Интервью с Ауке Джилдерда 145

II. Цели 157

Глава 9. Создание командного духа
Интервью с Гради Бучем 161

Глава 10. Если вы провалились
Дженнифер Грин 173

Глава 11. Планирование
Интервью с Майком Коном 183

Глава 12. Как борцы против авторских прав захватили Мордор
Кори Докторою 203

Глава 13. В защиту Свободного мира
Интервью с Нилом Сигелем 211

Глава 14. Спасение жизней
Интервью с Тревором Филдом 223

III. Методы 235

Глава 15. Создание команд: сотрудничество и обучение
Джеймс Греннинг 239

Глава 16. Эффективные методы
Интервью со Стивом Макконнеллом 257

Глава 17. Воспоминания о проекте повышения продуктивности разработки ПО
Барри Бём и Мария Э. Пенедо 277

Глава 18. Создание космических кораблей
Интервью с Питером Глюком 293

Глава 19. Успешное техническое задание
Карл Е. Вигерс 311

Глава 20. Разработка в Google
Интервью с Алексом Мартелли 331

Глава 21. Команды и инструменты
Карл Фогель 345

Глава 22. Исследовательские команды
Интервью с Майклом Коллинзом 371

Глава 23. Команда HADS
Карл Ремер 379

IV. Препятствия 399

Глава 24. Плохой босс
Эндрю Стеллман 403

Глава 25. Знакомьтесь: процесс разработки
Нед Робинсон 425

Глава 26. Преодолевая препятствия
Интервью со Скоттом У. Амблером 443

Глава 27. Скорость или качество
Джоанна Ротман 457

Глава 28. Тяжко, да?
Марк Денович и Эрик Ренки 477

Глава 29. Стереотипы и нестандартные решения
Патрисия Энсворт 511

Глава 30. Обобщая мнение команды
Энди Орам 539

V. Музыка 557

Глава 31. Создавая музыку
Интервью с Тони Висконти 561

Участники проекта 579

Алфавитный указатель 593

Отзывы

«Прекратите жаловаться на коллег. Вместо этого дайте им и боссу прочесть „Идеальные команды“. Эта книга доказывает, что командная работа имеет потрясающий потенциал, который может реализовать и ваша компания.»

Скотт Беркун, автор бестселлеров «The Miths of Innovation» и «Making Things Happen»[1]

«Эти истории, этот опыт и наблюдения удивительно различны, что дает более глубокую и широкую картину, чем получилось бы у любого единичного автора. Где-то в этой книге вы точно найдете ответы именно на те вопросы создания и управления командами, которые вы ищете в данный момент, а также расширите свой взгляд на те проблемы, с которыми пока еще не сталкивались.»

Алекс Мартелли, Google

«В „Идеальных командах“ авторы не дают обычных рекомендаций, а отмечают свои успехи (и проблемы), связанные с работой в команде. Этим они дают необходимое представление о том, что может встретиться в вашей собственной карьере. Дженнифер Грин делится своими удивительными воспоминаниями о провале доткомовского проекта; Кеоки Андрус выказывает здоровое уважение к инновациям и творческому подходу как способам воодушевления команды; интересные рассказы Карла Вигерса и многих других касаются их опыта работы в команде. „Идеальные команды“ – увлекательное чтение, в отличие от большинства технических книг.»

Джим Броссо, автор книги «Software Teamwork: Taking Ownership For Success»

---

[1] Скотт Беркун «Искусство управления IT-проектами». – Пер в англ. – СПб.: Питер, 2010.

«Команды делают программы, и если команда работает неэффективно, то в игре по разработке программ вы, скорее всего, проиграете. Опыт, изложенный в этой книге, пригодится вам на многие годы.»

Скотт У. Амблер, мировой лидер в области практики Agile-проектов, IBM Software Group, автор книги «Agile Modeling»[1]

«Хорошие, плохие, ужасные и, конечно, идеальные. Вы узнаете о командах, которые уцелели, – неважно, как; о том, как создавали свои команды некоторые известные организации. Если вы хотите узнать, как создать команду или сделать ее эффективной, прочтите эту книгу. Истории в ней увлекательны и реалистичны.»

Йоханна Ротман, консультант, автор книг

«Все, кому нужно организовать совместную работу группы, начинают инстинктивно собирать чужие истории: вы быстро понимаете, как сложна ваша задача и как ограничен ваш собственный опыт. Если вам знакомо это чувство, то эта книга для вас. Это репортажи из окопов, рассказанные красноречиво, с чувством юмора, заставляющие как читателя, так и писателя на миг вспомнить о похожей ситуации и поежиться.»

Карл Фогель, разработчик открытого ПО

[1] Скотт Амблер «Гибкие технологии: экстремальное программирование и унифицированный процесс разработки». – Пер в англ. – СПб.: Питер, 2005.

Почему «идеальные команды»?

За много лет нам довелось поработать со многими командами, в разных компаниях и над большим количеством разных проектов. И в ходе написания книг, статей и постов в блогах о том, как приводить проекты к успеху, нас всегда мучил один вопрос. Нашей задачей, как представлялось, было составить рецепты «лучших» способов реализации проектов по разработке программного обеспечения: как планировать проект, как разрабатывать продукт и как обеспечить его надежную работу. Но чем больше мы писали и чем больше говорили с людьми, тем более спорной начинала казаться эта идея.

Сомнения начались уже после написания нашей первой книги, которая задумывалась как сборник практических советов по ведению успешных программных проектов. Мы провели обширные исследования, много экспериментировали и работали с данными реально существующих или существовавших проектов, чтобы найти эффективные методики планирования программных проектов, написания лучших кодов и тестирования программных продуктов. Мы отобрали те, которые казались нам наиболее действенными, и изложили их в книге, стараясь сделать ее как можно менее объемной. В последующие годы мы получили огромное количество невероятно положительных отзывов, многие говорили нам, что книга стала для них настольной.

Именно тогда все и начало идти не так.

Как ни смешно, мы попали в ловушку, о которой сами же писали в книге: стали смотреть на вещи слишком узко. Определили данный конкретный набор методик как «успешный». Мы никогда не собирались утверждать, что есть только один способ планирования проекта, его оценки, проверки кода или тестирования программного продукта. Но сейчас, много общаясь с разными разработчиками и программистами, мы обнаружили, что мнения по поводу нашей книги были довольно однозначными. Нам говорили: «А, Стеллман и Грин, создатели метода Wideband Delphi? Я всегда применяю его для оценки!» (Честное слово, нам это говорили!) Или (что гораздо неприятнее): «Вы пишете о технических требованиях, которыми я никогда не пользуюсь. Но я разрабатываю отличные программные продукты. Значит, вы не правы!»

Методика – это теория. Методики сухи. Можно рассуждать об их достоинствах и выдвигать гипотетические аргументы, будут ли они работать в той или иной ситуации. Мы и сами поломали немало копий в спорах о том, на каком этапе нужно собирать технические требования и как сделать проект относительно гибким, Agile-проектом[1]. По таким вопросам зачастую ведутся целые войны.

Но, если честно, это не самое важное в проектах. Принятие таких решений, несомненно, влияет на проект, и порой очень сильно. Однако гораздо важнее – *кто* у вас в команде: что они умеют и как хорошо у них получается работать вместе. Только тогда мы поняли, что методы, техника – только один из аспектов разработки хороших программных продуктов. И хотя именно этому аспекту мы посвятили большую часть своей работы и исследований, но далеко не сразу осознали, что зачастую он не играет ключевой роли.

И мы захотели помочь людям рассказать о своих проектах и узнать, чему научились другие. Мы собрали все рассказы о лучших методах – ведь мы писали именно о них и разбирались в этом лучше всего – и попытались объяснить в самых общих и необязывающих словах, как можно применить их для лучшего ведения своего проекта. Мы наивно полагали, что людям будет интересно услышать, чему мы научились в ходе написания книги и при реализации наших собственных проектов. Нам хотелось рассказать о ловушках, которых можно избежать, и предложить пути обхода этих препятствий. («Посмотрите, как проводится эффективная оценка!»)

Это был полный провал. Особенно мрачные воспоминания остались об одной встрече, хотя она была ненамного хуже других наших встреч. Нас пригласили провести презентацию в рамках серии неформальных встреч[2] в офисе крупной нью-йоркской интернет-компании. Мы начали свой рассказ, перечисляя проблемы, с которыми обычно встречаются команды разработчиков, и излагая способы, позволяющие, по нашему опыту, их избежать. На лицах разработчиков все отчетливее читалась скука, даже разочарование. В основном это были молодые люди до трид-

[1] Agile (произносится «эджайл») – гибкая методология разработки ПО, где во главу ставятся принципы самоорганизации команды разработчиков, тесного личного взаимодействия между разработчиками и конечным пользователем, разумное сокращение документирования процессов, короткие итерации, позволяющие гибко реагировать на изменения в запросах заказчика. – *Прим. перев.*

[2] Brown bag series – распространенный формат встреч и обсуждений «за ланчем», принесенным в бумажных пакетах из дома или кафе, получивший соответствующее название. – *Прим. перев.*

цати, и к середине нашего рассказа большинство из них уже уткнулись в свои КПК и ноутбуки. Некоторые даже встали и ушли. В конце, когда мы подошли к ответам на вопросы слушателей, мы поняли, о чем они думали. Один программист, крупный мужчина с длинной седой бородой, с вызовом заявил нам, когда мы заговорили о планировании Agile-проектов: «В Agile не нужно ничего расписывать, садишься и пишешь код». И все в зале закивали. Мы пытались переубедить их, мол, концепция Agile совсем не обязательно сводится к этому, но они не слушали. Нам было до них не достучаться, мы все только зря теряли время.

Не стоило удивляться такому прохладному (в лучшем случае) приему. Да мы и сами прошли через кучу таких совещаний и встреч (группы по управлению проектом, SIG-группы по вычислительной архитектуре, координация разработки программного обеспечения...). И на каждую по-настоящему запоминающуюся встречу или обсуждение приходилась чуть не дюжина тех, что сводились к рекламе консалтинговой компании, книги, профессионального продукта или услуги и абсолютно нас не трогали.

Мы поразмыслили над замечанием по поводу принципов Agile, услышанным на встрече. И с удивлением поняли, что тот человек имел право упрекать нас, хотя фактически мы не могли с ним согласиться. Да, поначалу критика ощущалась весьма болезненно, и нам действительно понадобилось время, чтобы понять, почему та встреча закончилась столь плачевно. Но в итоге именно это замечание показало, в чем была наша ошибка. *Не существует лучшего метода* – по крайней мере, известного нам и гарантирующего неизменный успех. Будет ли работать тот или иной инструмент или техника, зависит от обстоятельств: проекта, людей и целой беспорядочной кучи факторов, которые невозможно просчитать или предусмотреть. Замечу, мы не хотим этим сказать, что нам не важны процесс, методология. Мы не открещиваемся от методов, которыми пользуемся сами и о которых пишем. Наоборот: мы по-прежнему применяем их в ежедневной работе, пишем о них и по-прежнему учим других разрабатывать лучшие программные продукты, пользуясь ими, потому что они действительно могут работать в правильной ситуации, для правильного проекта с правильными людьми. Но иногда, чтобы добиться чего-то, нужно действовать *как угодно*, но только не «правильно». Иногда годится любой действенный метод.

Понимание того, что есть много способов реализации проектов, способствовало нашему росту. Хотя мы нашли методы, в которых были уверены, в некоторых случаях они не работали. С годами нам пришлось стать гораздо более гибкими в отношении собственных подходов к разработке программного обеспечения и к работе с нашими командами. И причи-

ной тому стал провал некоторых наших проектов. Таких было немного, но они были. И именно эти неудачи стали лучшей школой для нас.

Это привело нас на новый путь. Мы попытались понять, почему некоторые проекты работали, а другие ждал провал. Поначалу было сложно определить конкретную причину, и мы начали думать, что однозначного ответа тут не найти. Но когда мы начали общаться с окружающими, со своими знакомыми, мы раз за разом слышали одно и то же: люди с любовью говорили о хороших проектах, которые им запомнились, хотели поделиться рассказами об ужасных руководителях, о тяжелых ситуациях, с которыми они столкнулись, или о проблемах, которые им удалось решить. Фактически вопрос о командах, в которых они работали, приводил людей к размышлениям об их собственной карьере и о том влиянии, которое они оказали на своих окружающих. К нашему удивлению, это оказалось свойственным всем специалистам, работающим на всех уровнях разработки программных продуктов, – от младших программистов, пару лет назад закончивших вуз, до технических директоров, прошедших долгий путь наверх, начав с должности младшего программиста.

Чем больше мы позволяли людям говорить об их опыте, не навязывая собственное мнение или наши методы, тем меньше споров у нас возникало. И чем больше мы говорили об этом между собой и с другими, тем более четко эти беседы делились на четыре «блока»: *люди* (кто в команде), *цели* (что их обединяет), *методы* (как они разрабатывают программы) и *препятствия* (что стоит на их пути). Эта книга разбита на части на основе этих идей. Завершает книгу глава о тех, кто занимается музыкальными проектами, потому что благодаря им мы лучше поняли, что способствует успешной работе коллектива.

Почему именно эти авторы?

Этот проект основан на постулате: чем больше вы знаете о том, как разные люди ведут свои проекты, тем лучше подготовитесь к ведению собственного. Мы начали с того, что обратились к своим знакомым, людям из наших команд и тем, с кем мы познакомились на конференциях или как-то пересекались в сфере разработки программных продуктов. Поразил разброс мнений, даже только среди наших знакомых. Нас удивляло, что не со всем сказанным мы могли согласиться, но, что важнее, мы тем не менее видели зерно истины в этих идеях.

Так и появилась связующая цель книги: собрать как можно больше различных мнений в разных областях разработки независимо от того, соответствуют ли они нашему мнению или друг другу. Да, это почти

наверняка означало, что мнения будут противоречивыми. Но мы решили, что это не только не проблема, но и преимущество. (It's a feature, not a bug.) Это доказывало, что хотя идеи и противоречивы, они могут работать в разных ситуациях. Все проекты уникальны, и все команды разные. Было бы странно, если бы существовал единый рецепт. Все эти мнения имеют право на жизнь, и бывают ситуации, когда любое из них может стать верным выходом.

Мы не хотим сказать, что не бывает неверных мнений. Наоборот, поэтому мы включили в эту книгу и рассказы о командах, которые *не могут* добиться успеха. Правильное решение в нужный момент может значительно повлиять на успех проекта, и люди могут допустить серьезные ошибки при разработке программ. Они выбирают неверную методику или путь: например, хорошая система документации может оказаться плохой, если она затруднительна в использовании и попросту не применяется. Однако в проектах может возникнуть и другая крайность, когда команда вообще ничего не планирует и каждый тянет в свою сторону. Вот что показано в предостерегающих историях этой книги: команды, которые поставили неверные цели, набрали не тех людей, применяли неправильные методы или просто встретили непреодолимое препятствие.

Самим нам никогда не удалось бы написать такую книгу. Наш опыт и знания очень ограничены, как у любого человека. И потому мы развернули широкую сеть, разговаривая с теми, чье мнение мы уважаем. Множество людей, которые приняли участие в создании этой книги, представляют собой реальный спектр проектов, которые ведутся по всей отрасли. И хотя многие из мнений и идей, о которых вы прочтете, не совпадают с нашим мнением – в некоторых случаях они даже противоположны, – мы многому научились у каждого из этих людей. Думаем, что научитесь и вы.

В книгу вошли истории и интервью с опытными руководителями по всей отрасли разработки программных средств. Мы старались подбирать участников, имеющих отношение к возможно большему количеству разных сфер и областей интересов – от обороны до социального устройства, от научных исследований до разработки видеоигр, от аэрокосмической промышленности до поисковых машин и от менеджеров проекта до рядовых программистов и системных администраторов. Это люди, с которыми мы познакомились в ходе учебы и работы. Участники представляют широкий спектр компаний, включая тех, кто работал (а в некоторых случаях и продолжает работать) на NASA, Google, IBM и Microsoft, чьи книги и лекции помогли нам понять, как разрабатываются программные продукты. Для нас было честью работать с ними

и с другими участниками этой книги, которых вы, возможно, также не знаете, но они делают удивительные и инновационные вещи в области разработки. Мы приносим особенную благодарность Тиму О'Рейли (чей издательский дом напечатал эту и другие наши книги) и Скотту Беркуну (с ним мы знакомы много лет, и он не только написал для нас замечательное эссе, но и взял интервью у Стива Макконнелла и Тони Висконти, легендарного музыкального продюсера, который нашел время рассказать нам о своих методах).

Если честно, мы до сих пор поражены, что нам удалось собрать такую широкую, компетентную, знаменитую и талантливую группу авторов. Но что еще удивительнее, никто из них не потребовал оплаты. Вместо того чтобы разделить гонорар за эту книгу между авторами, мы имеем возможность передать его в PlayPumps International, инновационный благотворительный фонд, занимающийся бурением скважин для поставки питьевой воды в школы и деревни Африки южнее Сахары. PlayPumps – это не просто благотворительная организация; им довелось заниматься проектированием и техническим усовершенствованием. Вы сможете больше узнать о них и об их миссии (а также немного о командах!) из нашего интервью с Тревором Филдом, основателем PlayPumps.

Каждому из этих авторов есть что сказать интересного, важного и, что еще весомее, *полезного* о командах: как они работают, как их собирать и почему они распадаются. Каждого из авторов с полным правом можно назвать опытным руководителем команды, познавшим как успех, так и неудачу. Иногда нас удивляли и даже шокировали их истории. Но каждая из них увлекательна. В сумме все эти элементы дают опыт, который не смог бы самостоятельно накопить ни один человек.

Прежде чем углубиться в четыре области, которые мы решили рассмотреть, – люди, цели, методы и препятствия, – хотим привести интервью с Тимом О'Рейли, потому что в нем затрагиваются темы, которые вы вновь и вновь встретите на страницах этой книги. Читая его, следите за каждой из четырех тем. В разговоре с нами Тим смог конкретизировать многие из этих понятий и облечь в слова многие идеи, зародившиеся у нас после редактирования историй и интервью в книге. Мы не знали, чего ожидать от интервью с Тимом О'Рейли. Из нас двоих Дженни встречалась с ним лишь раз на конференции Amazon Developer Days (куда ее пригласили с докладом), но мы читали его интервью и его статьи в O'Reilly Radar. В частности, нас интересовало, что он скажет о руководстве и о том, как управлять группами людей, не контролируя каждый их шаг.

Если вы надеетесь найти в этой книге *Единственный Верный Способ*™ получить идеальную команду, то нам очень жаль, потому что книга не об этом. Но если вы хотите получить представление о том, что именно

позволяет хорошим командам работать как часы и что можно сделать, чтобы средняя команда начала работать лучше – или чтобы сильная команда провалила проект, – эта книга сможет дать вам многое. Думаем, ее будет интересно читать, местами просто захватывающе интересно. В ней заключен огромный опыт множества людей со всей отрасли (и даже из других областей). Мы попытались определить общие принципы, которые сплачивают команды, не навязывая вам предписания, догмы, теории и прямые советы. Надеемся, вам понравится читать «Идеальные команды», как нам нравилось писать эту книгу.

Эндрю Стеллман и Дженнифер Грин

Март 2009 года

Предисловие

Книга «Идеальные команды» была задумана в конце 2007 года на встрече в кабинете редактора издательства O'Reilly Энди Орама в Кембридже (Массачусетс). Нас пригласили прочитать лекцию в местном отделении Института управления проектами, и мы решили воспользоваться возможностью и заглянуть в офис O'Reilly поздороваться и наконец-то своими глазами увидеть обладателей очень знакомых голосов, с которыми мы работали много лет. «Beautiful Codes»[1] несколько месяцев возглавлял список бестселлеров издательства, и компания собиралась выпустить следующую антологию. Поскольку большую часть своей карьеры мы читаем лекции и пишем книги о командной работе над проектами, то идея книги «Идеальные команды» висела в воздухе.

Сначала предполагалось, что книга, подобно «Идеальному коду», будет подборкой статей о руководстве проектами. Но, как и все великие проекты, «Идеальные команды» зажили собственной жизнью. В написании приняли участие мастера высочайшего класса. Для нас это стало увлекательным путешествием: мы получили возможность научиться чему-то у самых ярких индивидуальностей в современном программировании. Это истории из личного опыта. Каждый из авторов этой книги говорит о собственной работе, а подобный опыт есть у очень немногих из нас. И все авторы рады были поделиться своим временем и усилиями бесплатно: доходы от этой книги будут направлены в PlayPumps International.

Как организована книга

Вот перечень глав и их краткое содержание:

Глава 1 «Лидерство» (интервью с Тимом О'Рейли)

> Тим О'Рейли делится мыслями о ведущих командах и компаниях и о прогрессе мира программирования.

Часть I. Люди

Глава 2 «Почему побеждают ужасные команды» (Скотт Беркун)

> Многие из тех, кто работает в так называемых идеальных командах, никогда их так не называют. Скотт рассказывает о своей работе в компании Microsoft и объясняет эстетику ужасных команд.

---

[1] Э. Орам, Г. Уилсон «Идеальный код». – Пер в англ. – СПб.: Питер, 2009.

Глава 3 «Создание видеоигр» (интервью с Марком Хили)

Один из основателей Media Molecule рассказывает о том, чему он научился при создании популярной видеоигры LittleBigPlanet.

Глава 4 «Построение идеальной команды» (Билл Дипьер)

Билл пишет, как хороший менеджер может сделать великую команду из разрозненной группы.

Глава 5 «Что мотивирует разработчиков» (интервью с Энди Лестером)

Менеджер-программист, участник создания Perl, Энди отвечает на вопросы о том, что мотивирует разработчиков и как можно улучшить взаимоотношения в команде.

Глава 6 «Как воодушевлять людей» (интервью с Кеоки Андрусом)

Кеоки рассказывает о том, как ему удалось улучшить команды в таких компаниях, как Intuit, Microsoft и Novell. Ключ к успеху – понимание, воодушевление людей и умелое управление ими.

Глава 7 «Музыкальную индустрию – в XXI век. Одна тяжба за раз» (Том Тарка)

Это история о взлете и падении MP3.com, иконы бума доткомов, а также о тех, кто это пережил.

Глава 8 «Inner Source – ПО с исходным кодом, открытым для внутреннего использования» (интервью с Ауке Джилдерда)

Идея Inner Source состоит в том, что принципы программирования с открытым исходным кодом внедряются в корпоративные команды. Ауке отвечает на вопросы о том, как он применял их и как это повлияло на людей в этих командах.

Часть II. Цели

Глава 9 «Создание командного духа» (интервью с Гради Бучем)

Чтобы сплотить команду, нужно приложить усилия, особенно если сотрудники работают над разными задачами. Гради рассказывает о трудностях на пути движения команд в верном направлении.

Глава 10 «Если вы провалились» (Дженнифер Грин)

Дженнифер повествует о том, как она работала в идеальной команде с конфликтующими задачами.

Глава 11 «Планирование» (интервью с Майком Коном)

Примерами из своей карьеры Майк иллюстрирует то, каким образом понимание общего контекста проекта определяет разницу между успехом и неудачей.

Глава 12 «Как борцы против авторских прав захватили Мордор» (Кори Доктороу)

Это история о том, как выдающаяся команда, преисполненная ответственности перед обществом, смогла победить страшного врага.

Глава 13 «В защиту Свободного мира» (интервью с Нилом Сигелем)

Технический директор крупной оборонной и аэрокосмической компании рассказывает о том, как он мотивирует своих разработчиков.

Глава 14 «Спасение жизней» (интервью с Тревором Филдом)

Основатель PlayPumps International вспоминает о том, что побудило его оставить гламурный мир рекламы и посвятить свою жизнь доставке чистой питьевой воды в сельские школы и деревни Африки к югу от Сахары.

Часть III. Методы

Глава 15 «Создание команд: сотрудничество и обучение» (Джеймс Гренннинг)

Один из Agile-пионеров и издателей Agile Manifesto, Джеймс рассказывает о своих первых опытах применения Agile-методов.

Глава 16 «Эффективные методы» (интервью со Стивом Макконнеллом)

Скотт Беркун и Стив Макконнелл обсуждают то, каким образом лучшие методики разработки могут привести к большей производительности.

Глава 17 «Воспоминания о Проекте повышения продуктивности разработки ПО» (Барри Бём и Мария Э. Пенедо)

Это история одного из первых улучшений процесса, рассказанная пионерами индустрии.

Глава 18 «Создание космических кораблей» (интервью с Питером Глюком)

Питер делится воспоминаниями о проблемах создания программ для космоса в Лаборатории реактивного движения НАСА.

Глава 19 «Успешное техническое задание» (Карл Е. Вигерс)

Требования к программам могут спасти или погубить проект, и Карл использует их для обеспечения успеха.

Глава 20 «Разработка в Google» (интервью с Алексом Мартелли)

Алекс говорит о том, как удачное планирование и Agile-методики помогают работать лучшим компаниям.

Глава 21 «Команды и инструменты» (Карл Фогель)

Карл показывает, как инструмент программирования может оказать огромное воздействие на работу компании.

Глава 22 «Исследовательские команды» (интервью с Майклом Коллинзом)

Майкл рассказывает о своей работе над проектом по исследованию безопасности.

Глава 23 «Команда HADS» (Карл Ремер)

Работа над программным обеспечением полетов для Боинга-777 потребовала написания принципиально нового набора инструментов, что привело к новым проблемам.

Часть IV. Препятствия

Глава 24 «Плохой босс» (Эндрю Стеллман)

Один плохой руководитель может развалить команду.

Глава 25 «Знакомьтесь: процесс разработки» (Нед Робинсон)

Хорошей команде по плечу самые невероятные и непредвиденные задачи!

Глава 26 «Преодолевая препятствия» (интервью со Скоттом У. Амблером)

На пути команды могут возникать самые разные препоны. Скотт рассказывает о том, как преодолеть самые крупные из них.

Глава 27 «Скорость или качество» (Джоанна Ротман)

Новый руководитель проекта сталкивается с рядом проблем.

Глава 28 «Тяжко, да?» (Марк Денович и Эрик Ренки)

Невероятно крутая команда преодолевает одну проблему за другой.

Глава 29 «Стереотипы и нестандартные решения» (Патрисия Энсворт)

Команда при плохом руководстве, в ужасных условиях и с межличностными проблемами тем не менее выживает.

Глава 30 «Обобщая мнение команды» (Энди Орам)

Один разработчик может принять на себя руководство, если факты складываются в его пользу.

Часть V. Музыка

Глава 31 «Создавая музыку» (интервью с Тони Висконти)

Размышления легендарного продюсера звукозаписи Тони Висконти о том, что у продюсирования музыки и создания программ много общего.

Контактная информация

Отзывы и вопросы по книге, пожалуйста, отправляйте в издательство:
O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
800-998-9938 (в США и Канаде)
707-829-0515 (международный или местный)
707-829-0104 (факс)

У этой книги есть веб-страница, где мы указываем опечатки, примеры и другую дополнительную информацию. Адрес страницы:

http://www.oreilly.com/catalog/9780596518028

Для комментариев и технических вопросов о книге отправляйте электронные письма по адресу:

bookquestions@oreilly.com

Чтобы узнать больше о наших книгах, конференциях, ресурсных центрах и Сети O'Reilly, посетите наш веб-сайт:

http://www.oreilly.com

Safari Books Online

Значок Safari® Books Online на обложке вашей любимой книги о технологиях означает, что книга доступна в Интернете через O'Reilly Network Safari Bookshelf.

Safari – это лучшее решение, чем электронные книги. Это виртуальная библиотека, которая позволяет легко искать по тысячам лучших технических книг, копировать и вставлять образцы кодов, скачивать главы и находить быстрые ответы, если вам нужна самая точная и новая информация. Бесплатно оцените эту систему на *http://my.safaribooksonline.com*.

Благодарности

Мы хотели бы поблагодарить всех наших соавторов, великодушно вложивших свои время и силы: Тима О'Рейли, Скотта Беркуна, Марка Хили (и всех сотрудников Media Molecule!), Билла Дипьера, Энди Лестера, Кеоки Андруса, Томми Тарка, Ауке Джилдерда, Гради Буча, Майка Кона, Кори Доктороу, Нила Сигеля, Джеймса Греннинга, Стива Макконнелла, Барри Бёма, Марию Пенедо, Питера Глюка, Карла Вигерса, Алекса Мартелли, Карла Фогеля, Майка Коллинза, Карла Ремера, Неда Робинсона, Скотта Амблера, Джоанну Ротман, Марка Деновича, Эрика Ренки, Стэна Гранайта, Патрисию Энсуорт и Энди Орама. Большое спасибо Тони Висконти за то, что он нашел время с нами поговорить.

Мы хотим поблагодарить Тревора Филда и других сотрудников PlayPumps International (*http://www.playpumps.org*) за их замечательную работу, а также за вклад Тревора в эту книгу. Пожалуйста, посетите вебсайт этой организации, чтобы узнать, почему мы направляем им доход от этой книги.

И наконец, особая благодарность Нише Сондхе за прекрасные фотографии.

Об авторах

Эндрю Стеллман начал работать программистом в национальном звукозаписывающем лейбле после окончания Школы компьютерных наук в Университете Карнеги-Меллона. Первые несколько лет он специализировался на техническом дизайне и архитектуре, но проявил себя как очень ответственный лидер и руководитель команд, в которых работал. Последние десять лет Эндрю руководил командами программистов и многочисленными проектами. Сейчас он работает над собственными проектами и помогает улучшать проекты другим.

Дженнифер Грин начала карьеру инженера-программиста в начале 1990-х в известном онлайн-сервисе. С тех пор она создавала и возглавляла команды инженеров-программистов для организаций, специализирующихся на финансах, некоммерческой деятельности, работе с естественным языком и медиа. Как создатель и руководитель рассредоточенных команд разработчиков она работала с программистами всего мира, делясь своим опытом и помогая улучшать программы.

Дженнифер и Эндрю работают вместе с 1998 года, создавая программы и тренируя команды разработчиков. Сначала они координировали процесс разработки в своей компании, а впоследствии осуществили множество успешных совместных проектов. Их проекты ориентированы на науку и здравоохранение; они создали программное обеспечение для Медицинской школы Мэйлмана (Колумбийский университет), Бизнес-школы (там же) и Национальной академии наук. Затем они перешли к коммерческому руководству проектами и консультированию программистов: занимались тренингом, проект-менеджментом и удаленным проект-менеджментом, оказывая профессиональные услуги деловому и академическому сообществам.

Их первая книга «Applied Software Project Management» (Проект-менеджмент в области прикладного программирования) была опубликована в издательстве O'Reilly в 2005 году и стала базовым учебником по менеджменту в программировании во многих университетах мира. За-

тем они написали еще две книги для знаменитой серии «Head First». «Head First PMP» (Проект-менеджмент для начинающих) стала одним из лучших учебников по руководству проектами, ее используют десятки тысяч менеджеров проектов для подготовки к сертификации. Третья их книга «Head First C#» больше года была в списке бестселлеров издательства O'Reilly и остается одной из самых продаваемых книг о языках программирования. Помимо писательской деятельности Эндрю и Дженнифер регулярно выступают на конференциях по всему миру по вопросам проект-менеджмента и улучшения процесса разработки. «Идеальные команды» – их четвертая книга.

1

Лидерство

Интервью с Тимом О'Рейли

Тим О'Рейли основал компанию, которая выросла потом в O'Reilly Media, Inc., – она и напечатала эту книгу (а также другие наши книги). Мы всегда восхищались им самим и компанией, которую он создал, и не только как издателем, но и в плане общего влияния на мир программного обеспечения и его разработки. В течение многих лет нам довелось поработать с людьми из почти всех подразделений O'Reilly, и хорошую команду мы всегда распознаем. Мы хотели, чтобы Тим поделился тем, как ему удалось создать такой коллектив и постоянно поддерживать высокую эффективность его работы.

Дженни: **Начав собирать материал для этой книги, мы обратили внимание, что все воспринимают понятие «команда» несколько поразному. Оказалось, что в некоторых из отобранных нами для книги историй оно получает однозначное толкование: либо команда — это обязательно люди, которые временно работают вместе над конкретным проектом, либо командой может быть группа людей, которые никогда не встречались.**

Эндрю: **Мы решили немного схитрить и оставить этот вопрос открытым для разных интерпретаций. А как бы вы определили понятие «команда»?**

Тим: Я в основном занимался руководством компанией. Да, сюда входит и опыт работы с командой. Но мои размышления на эту тему скорее связаны с более широким вопросом реализации лидерских способностей. Давайте начнем с лидерства и руководства, которые тоже входят в концепцию «команды». Я сам не вполне понимаю, где между ними граница.

Хочу поделиться с вами двумя цитатами. Одна из них принадлежит Гарольду Дженину, основавшему ITT, по сути первую многопрофильную корпорацию-конгломерат. Он сказал: «Искусство руководителя заключается в умении достигать своих целей силами других людей». Довольно интересная концепция. Вопрос здесь в разных способах ее реализации, и некоторые из них очень «директивны». Классический «менеджер» представляется как человек, определяющий задачи работы, людей, которые должны ее выполнить, и организующий коллективы с распределением ролей, и прочее в этом духе.

Я полностью поддерживаю эту концепцию, потому что умение руководить действительно представляет собой умение достигать своих целей за счет усилий других. Однако я всегда предпочитал придерживаться доктрины, сформулированной в другой цитате, речь в которой, на самом деле, идет о писательском мастерстве. Принадлежит она Эдвину Шлоссбергу, автору журнальной статьи, которую я прочел еще на заре своей карьеры. Его слова так запали мне в душу, что стали для меня своего рода основополагающей идеей в дальнейшей работе. Звучит она так: «Искусство писателя заключается в умении создать контекст, в рамках которого другие смогут думать».

Дженни: **По-вашему, можно применить эту идею — «создание контекста, в рамках которого другие смогут думать» — к командам разработчиков?**

Тим: Я вам расскажу о том, что я называю «архитектурой участия». В 1998 году мы готовили книгу под названием «Open Sources» (ПО с открытым исходным кодом) и брали интервью у тех, кто уже писал об этом. И Линус Торвальдс сказал в одном интервью то, что мне запом-

нилось, хотя и не думаю, что это попало в книгу. Он сказал: «У меня не получилось бы сделать для Windows то, что я сделал с Linux, даже если бы у меня был доступ к исходному коду Windows. Просто у нее другая архитектура». В результате я задумался об архитектуре проектов с открытым кодом и о том, как их спроектировали таким образом, чтобы позволить подобную импровизацию в произвольном формате. Потому что эти правила уже были сформулированы.

Эндрю: **Забавно слышать о том, как Линус сказал, что он чего-то не может — хорошее напоминание о том, что в SourceForge.net полно проектов с открытым исходным кодом, которые так и не были реализованы.**

Тим: Мне кажется, некоторые проекты могут проваливаться и из-за того, что они смешивают и стыкуют несовместимые системы. Необходима система с фундаментально заложенной *возможностью* независимого использования отдельных ее элементов. Думаю, именно поэтому, например, wiki-книги так и не стали популярными. Почему? Потому что книга – это достаточно большая, сложная вещь с единой нитью повествования. А Википедия – это набор страниц, и по каждой смысловой единице кто-то может заявить правдоподобную информацию, а другие вольны ее обновлять и дополнять. И общее целое – это сумма великого множества таких маленьких частей. Думаю, что определенные виды работ позволяют применять подобную совместную деятельность – будучи свободными по форме именно потому, что они проектировались с учетом хорошей сочетаемости отдельных элементов.

ОС UNIX спроектирована подобно конструктору Лего, где принципы разработки предполагают наличие стыкующихся выступов и пазов. Все эти каналы, фильтры и прочее – люди могут писать совершенно независимые утилиты и знать, что они подойдут. Большинство программ считывают стандартный формат и выдают стандартный формат, правил немного и они просты.

Для каждой составляющей есть страница руководства (manpage). Если бы у меня был выбор между Windows с открытым исходным кодом под, скажем, общедоступной лицензией (GPL), или Windows с руководством по каждому маленькому отрезку кода, то я бы всегда думал: что можно выгрузить и что изменить? Да, у вас есть лицензия, но ее одной недостаточно. С другой стороны, руководств *было бы* достаточно, хотя и нарушались бы авторские права нескольких человек.

Эндрю: **То есть вы хотите сказать, что секрет успешного ведения проекта — с открытым кодом или любого другого — заключается в разбивке его на составные части, в которых люди смогут разобраться?**

Тим: Я говорю о том, что есть рамки, в которых вы организуете творческую работу людей. И здесь работает целая куча принципов. Расскажу

одну историю из 1998 года, когда я организовывал мероприятие, позже названное Open Source Summit. Она показывает, что происходило на тот момент в отрасли. Об этом, в частности, писал Эрик Рэймонд в книге «The Cathedral and the Bazaar»[1] («Собор и Базар»). Годом раньше я провел конференцию по Perl. Все мы параллельно думали об одних и тех же вещах. Netscape только что открыла исходный код своего навигатора, что вызвало большое брожение умов. И вот что я заметил: каноническая версия, изложенная Эриком Рэймондом и другими, полностью упускала из виду BSD[2]. Она рассматривала только Linux, только ПО по общедоступной лицензии, только идеалы свободного программного обеспечения.

Между тем, я видел и совсем иную традицию. Я посмотрел на нее и понял: «Ого, да эта традиция куда влиятельнее». И я, например, стал предлагать всем назвать пять главных программ в Интернете. Они чесали в затылках, а я говорил: «Номер один – BIND (Berkley Internet Name Daemon)[3]. У каждого из вас есть веб-сайт, который зависит от этой программы, поддерживаемой этим вот лохматым программистом из Редвуд-Сити (Калифорния). Номер два – sendmail, или Apache. 75% электронной почты в Сети маршрутизируется при помощи sendmail». Мы продолжали по списку, и все эти программы были разработаны в Беркли. Люди этого просто не видели.

Дженни: **Я помню, в то время была большая полемика по поводу выбора лицензии для открытого кода, GPL или BSD. Похоже, вам удалось стать выше всего этого, объединив этих людей. Как у вас это получилось?**

Тим: Я собрал сторонников GPL и тех, кто поддерживал лицензию Беркли, и сказал им: «Нам нужно понять, что у нас общего. Нужна единая картина. В 17.00 у нас пресс-конференция. Не знаю, что мы будем говорить, но в пять нам придется им что-то сказать». Я был уверен, что произойдет нечто интересное. Но я понятия не имел, к чему придет в течение дня эта группа из 20 человек, воюя по каждому вопросу. Пришел

---

[1] Эссе Эрика Рэймонда (Eric Raymond) «Собор и базар» в русском переводе было опубликовано в №№ 9 и 10 журнала «Открытые системы» за 1999 г. – *Прим. перев.*

[2] BSD (Berkley Software Distribution) – система распространения ПО в исходных кодах на основе ОС UNIX, разработанная и распространяемая сотрудниками Калифорнийского университета в Беркли в 1977–1995 гг. На базе BSD-UNIX было создано несколько операционных систем. – *Прим. перев.*

[3] BIND (Berkley Internet Name Daemon, или Domain) – DNS-сервер, обеспечивающий выполнение преобразования DNS-имени в IP-адрес, и наоборот. – *Прим. перев.*

Эрик и сказал: «У нас было совещание пару недель назад, и Кристина Петерсон предложила название „Open Source“». Майкл Тиманн из Cygnus заявил: «Мы тоже над этим думали, над проблемой названия для „свободного ПО“». Линус Торвальдс сказал, что он и не задумывался, что слово free (свободный; бесплатный) имеет в английском два значения. А Майкл Тиманн ему: «А мы называли это sourceware».

Было голосование. И мы решили: общими силами придумываем название и все начинаем им пользоваться. Мы выбирали между sourceware и open source, и второе победило. В пять мы провели пресс-конференцию, что было дальше, известно всем.

Эндрю: **Интересно, мы с Дженни как раз писали о том, что постановка придуманных дедлайнов может негативно отразиться на команде. А в вашем случае это, кажется, сработало.**

Тим: Главное, что мы сделали, – собрали кратковременную группу. Я чувствовал, что это может сработать, но не знал точно, во что это выльется или должно вылиться. Но я точно знал, что хочу, чтобы что-то произошло. То есть один из принципов работы команды говорит о том, что создание искусственного дедлайна: «Ребята, у нас пресс-конференция в пять», – может быть очень мощным инструментом. Однако многие компании и люди неправильно им пользуются. Я видел ситуации, где менеджеры проектов врали своим подчиненным о сроках сдачи работы, пытаясь искусственно создать ощущение срочности. Это разрушает доверие.

Но если подойти к этому правильно, подчеркивание такого чувства срочности может отлично сработать. Это немного сродни тому, что Александр Поуп сказал по поводу рифмованных двустиший: узкие рамки способствуют созданию напора в творчестве, оно бьет, как струи воды в фонтане.

В отношении команд, чего я всегда старался придерживаться – это отбора в команду людей с дополняющими друг друга способностями. В настоящий момент в O'Reilly очень сильный исполнительный директор, роль которого просто огромна. Мы с ней как две стороны одной монеты. Я работаю скорее с масштабными вопросами и трачу куда меньше времени на всякие внутренние дела, что позволяет мне делать гораздо больше в своей сфере. А она работает именно с ежедневными деловыми вопросами, и у нее это получается гораздо лучше, чем получалось у меня, когда я пытался совместить оба эти занятия. Четкое понимание дополняющих друг друга компетенций очень важно.

Эндрю: **У меня вопрос. Какое-то время назад мы с Дженни подготовили материал для портала ONLamp под названием «Чему корпоративным проектам стоит поучиться у ПО с открытым кодом». После этого**

мы провели лекцию на тему «За счет чего работает свободное ПО». Мы, в основном, рассматривали используемые ими методы, потому что тогда это был наш конек. Вы много говорите о командах, личностях, дополняющих компетенциях, о том, как работать с людьми. Но мы обнаружили вот что: если взять самые успешные проекты ПО с открытым кодом, хорошо всем известные и любимые – такие как Apache, Linux, Emacs, – обнаруживаешь множество очень хороших методик. Разработчики, программисты, участники проектов добровольно принимали методики, которые, работай они в офисе, показались бы им сильно ограничивающими их свободу, хотя это то же самое. Например, очень жесткий регламент разработки и выпуска, непрерывная интеграция, разработка с учетом тестирования.

Дженни: **Постоянные проверки кода. Нам дали задание внедрить те же методы в условиях корпоративной работы, и мы встретили сильное сопротивление со стороны тех же самых людей. Как вы думаете, почему?**

Тим: Попробую это сформулировать. Любой, кто ездил на лошади, знает, что секрет успеха в том, чтобы лошадь думала, что она делает то, что хочет. Поэтому, мне кажется, когда люди чувствуют, что им диктуют, что нужно делать, сопротивление неизбежно. Но если им кажется, что это их собственное желание, они делают это не задумываясь.

Две с половиной тысячи лет назад Лао Цзы сказал: когда людьми управляет лучший из руководителей, они говорят, что все сделали сами.

Когда создается что-то воистину чудесное с эстетической точки зрения, оно кажется очевидным. Теперь, когда появился iPod, как вообразить себе мир без сенсорных панелей? Помню, как впервые взял в руки электронную книгу (Kindle). Я начал тыкать в экран, и когда ничего не произошло, возникло недоумение – что-то не так? Теперь у Kindle есть, так сказать, свои неоспоримые истины – например возможность взаимодействия с EVDO[1], и уже думаешь: да, именно так это и должно работать.

Эндрю: **Интересно, что вы заговорили о том, как должен работать продукт. В большинстве случаев при обсуждении свободного ПО со мной говорили о таких вещах, как сравнение общедоступной лицензии GPL, лицензий BSD и Creative Commons[2], и мне кажется, люди зачастую не-**

---

[1] EVDO (Evolution Data Only) – технология скоростного мобильного беспроводного доступа в Интернет. – *Прим. перев.*

[2] Creative Commons – семейство лицензий Creative Commons, разрешающих более свободное использование творческих работ. – *Прим. перев.*

сколько увязают в таких деталях. Но вы говорите о ПО с открытым исходным кодом как об уникальном способе реализации определенной концепции, а я никогда не смотрел на него с этой точки зрения.

Тим: Да, необходимо понимать, что люди все время застревают на вопросе лицензирования, считая его критичным для свободного ПО. И хотя это действительно важный аспект, я думаю, что, как вы и предположили сейчас, практическая реализация и методики куда важнее. Если вспомнить историю нашего бизнеса, выпускаемые нами книги готовились в первую очередь при содействии пользователей. Большинство нашей продукции создается людьми, которые не работают на нас. Нам достаточно выдвинуть правдоподобную идею того, что нам хотелось бы видеть сделанным. Потом нужно найти кого-то, кто скажет: «А что, хорошая мысль, мне тоже это нужно». Мы даем разъяснения, обеспечиваем проверку их работы с помощью массы других людей внутри нашей компании и за ее пределами. Нам приходится внедрять систему управления, хотя зачастую она очень нестрогая. Есть много разных способов реализации этого, поэтому я так люблю слоган Ларри Уолла, создателя Perl: «Есть не один способ сделать это». Потому что я не считаю, что на каждый вопрос есть только один ответ.

Эндрю: **И особенно в Perl.**

Тим: Да. Взять книгу «Programming Perl»[1]. Я мог бы шесть месяцев работать над ее правкой и приведением в соответствие с моими представлениями об идеале. А мог сказать: «Ух ты, это уже отличная книга! Хотя лично я бы написал ее по-другому». И я дал добро и отправил ее в печать.

Помню одну изданную нами книгу. С ней все складывалось неудачно, потому что редактор просто зациклилась на переделывании ее во что-то совсем иное; два года она билась с авторами, пытаясь научить их, заставляя и увещевая, чтобы они написали все по-другому, чем в первоначальном варианте. Она просто не могла отступиться. И я сказал: «Смотри, вот у тебя книга, но ты себе представляла ее совсем по-другому». А ребята, надо сказать, просто свалили в кучу разнообразные несвязные истории. И я предложил: «Теперь просто напиши общую фабулу, и у книги появится смысл».

Дженни: **В командах, где я работала, очень многие говорили, что «Программирование на Perl» — одна из лучших в истории книг по программированию. Примерно представляешь себе, чего ждать от книги, а она оказывается совсем другой, даже лучше, чем ты думал.**

---

[1] Уолл Л., Кристианен Т., Орвант Дж. «Программирование на Perl». — Пер. с англ. — СПб.: Символ-Плюс, 2002.

Тим: Это еще одна из сторон. Как распознать, где и в чем истина? Мне кажется, в этом чуть ли не основа интеллекта, откуда выводится все остальное. Интеллект бывает разным. Одно из его проявлений, по сути, алгоритмично и манипулятивно: вам дается массив данных, и вы хорошо умеете ими управлять. Есть люди, замечательно умные в определенных вещах, но совершенно тупые во всех других отношениях. Они, например, могут не замечать очевидного или не дружить со здравым смыслом. И есть другие, которые плохо ориентируются в абстрактных понятиях, не умеют писать грамотно, безнадежны в математике. Но они гениальны в оценке ситуации и понимании природы происходящего вокруг. Самые умные могут совмещать эти качества. Они смотрят на мир свежим взглядом, с мыслью: «О, я знаю, что из этого можно сделать».

То есть, возвращаясь к той книге, у редактора было необходимое образование, представление о том, как должна выглядеть книга и каким образом все это должно работать. И это мешало ей посмотреть на материал свежим взглядом и понять, что работа может пойти, достаточно помочь авторам сделать то, чего они и сами хотят.

Дженни: **Вы считаете, что редактор — или, по сути, любой руководитель группы — не должен пытаться твердой рукой вести авторов (или свою команду) к определенной цели?**

Тим: Почти всю свою жизнь я был руководителем, при этом я не всегда был членом команды. В основном руководство заключается в том, чтобы направлять людей, и я стараюсь делать это, сводя свою роль к минимуму. Главное – видеть сильные стороны разных людей, видеть ситуацию и говорить: «Вот что мы можем из этого сделать».

Это как распознавание образа, и тут я возвращаюсь к своей позиции в отношении редакторской работы. Это в чем-то похоже на то, как Микеланджело описывал создание статуи: нужно увидеть образ, уже заложенный в камне. Мне кажется, редакторская работа сродни этому. И управление проектом сродни этому. Когда я начал писать о Web 2.0, моя работа заключалась в том, чтобы в массивах данных разглядеть эту скрытую в камне статую. То же было и с книгой про свободное ПО. Мне кажется, руководство командой происходит по тому же принципу. Как добиться, чтобы группа людей раскрыла свой потенциал? Только разглядев, кто они и чего они могут достичь.

Эндрю: **Как вы думаете, возможно ли существование великой команды без великого лидера? Команды, управляемой скорее коллективно?**

Тим: Да, это возможно. Но тут дело вот в чем. Возьмем, скажем, Apache, потому что, мне кажется, Apache – очень хороший пример такой команды. Тим Бернерс-Ли придумал рецепт. Он сказал: «Я создал идею гипертекстового сервера, гипертекстового клиента». Гениальность Apache

заключалась в том, что они умели обратить ограничения в достоинство. Я хорошо помню то время, середину 1990-х, когда Netscape добавила одно, Microsoft другое, и все говорили: что-то Apache никуда не двигаются. Не добавляют функции. Отстают! А ребята из Apache отвечали: «Ага. Ну, мы вообще-то делаем гипертекстовый сервер, и у нас есть отличный механизм, позволяющий каждому, кто желает чего-то нового, добавить его самостоятельно».

Тут мы снова возвращаемся к идее архитектуры участия. Они не строили крупное, сложносоставное приложение, а оставались в рамках простой концепции. Концепцию, на самом деле, придумал дальновидный лидер; она не была заложена в Apache изначально. Создала Apache группа людей, от которых откололась команда, работавшая над сервером NCSA, она ушла и основала Netscape. Но поскольку осталось определенное количество клиентов, эта группа решила: «Нужно продолжать поддерживать сервер». Самым чудесным в этой команде было то, что они соглашались с ограничениями, налагаемыми самой архитектурой их системы. Они не пытались выставлять напоказ себя и свою креативность.

Мне кажется, большая часть того, что делала IETF (the Internet Engineering Task Force – специальная комиссия интернет-разработки) в свои ранние годы, близка к этому. Были изложены замечательные принципы, и люди действительно их чтили. Если почитать, что писал Джон Постел в TCP RFC о принципах отказоустойчивости, это же как стих из Библии, право слово! «Будь консерватором в том, что делаешь; будь либералом в том, что перенимаешь у других». Это дословная цитата.

Суть в том, что если у вашей системы правильная архитектура, шансы вашей команды на успех значительно выше. Нежелательно иметь команду, целиком зависящую от чьего-то видения или от руководителя, потому что с потерей лидера развалится и команда.

Эндрю: **Говоря о провалах, есть ли в вашем прошлом истории, когда дела шли не очень хорошо? Может быть, расскажете о паре таких провалов, которые, возможно, научили вас чему-то важному?**

Тим: За все эти годы было немало провалов, некоторые из них впоследствии обратились в успех.

Это были не катастрофы, а неудачи. Это вопрос выбора, возвращаясь к теме лидерства. Дела могут пойти плохо. Посмотрите, что происходит с Yahoo! и Microsoft. Все вокруг говорят, что никак не выберут, кто же больший неудачник, Джерри Янг или Стив Балмер, потому что виной всему плохо организованный процесс без четкого видения верного пути развития. Но, в итоге, мы, можем быть, скажем и так: «Джерри был великолепен! Сохранил независимость Yahoo!, а потом придумал

стратегию, которая спасла их в последний момент». Вы же можете это представить? Если посмотреть на ранние годы создания ПК, на сделку, заключенную Биллом Гейтсом с IBM, – что если бы ловким дельцом там выступила IBM? История сложилась бы иначе. То есть мы просто не можем знать.

Дженни: **Это правда; то, что сегодня кажется неоспоримым успехом, в свое время могло казаться обычной глупостью. Вы последовательно делаете свой выбор, и будь что будет. Рассматриваете ли вы свои успехи и неудачи в таком ключе?**

Тим: В таких ситуациях я очень хорошо вижу, насколько все зависит от правильного выбора. Мы в O'Reilly создали первый веб портал. Это был GNN, Global Network Navigator, предшественник Yahoo!, первый портал сайтов Интернета. Я был сконцентрирован на другой работе и не хотел потерять контроль над своей компанией, поэтому мы продали GNN компании AOL. Я сделал правильный вывод, что не отдав в другие руки управление своей компанией и не привлекая инвесторов, мне не успеть за растущим Интернетом. Я прочел чудесную книгу «Marketing High Technology» (Маркетинг высоких технологий) Уильяма Давидова (Free Press), одного из первых венчурных инвесторов. Он говорит, это чистая арифметика. Чтобы занимать доминирующее положение на рынке, нужно занимать больше половины этого рынка и расти быстрее, чем рынок в целом. Я посмотрел на Интернет и сказал: «Не потянем, мы частная, полностью самофинансируемая компания. При той взрывной скорости, с какой развивается Интернет, мы неизбежно останемся на обочине. Надо либо привлекать средства, либо продавать». Я продал. Это была хорошая сделка для неопытного в этой сфере человека. А Джерри и Дэвид привлекли венчурные инвестиции и продолжили создание мультимиллиардных компаний невзирая на недавние трудности. Думаю, с чисто финансовой точки зрения они были гораздо успешнее нас. При этом я четко осознавал свои личные цели: сохранить независимую компанию, где я могу делать, что хочу, и именно этим я с тех пор и занимался.

То есть я пытаюсь сказать, что успех и провал всегда относительны, в зависимости от того, чего вы хотите достичь. Если бы мы были начинающей компанией, субсидируемой венчурным капиталом, мое решение было бы катастрофой. Как самофинансируемый предприниматель, который может решать сам за себя, я был волен поступить как мне угодно. Я веду к тому, что очень важно не предрекать заранее «провал», потому что на самом деле провал может оказаться вашим шансом.

Эндрю: **Но разве не нужно подводить какой-то итог? В конце концов, проект либо успешен, либо нет, разве не так?**

Тим: Это очень полярное мышление – либо успех, либо поражение. Мы выбираем себе будущее из всех возможных альтернатив. Нет просто лучшего, это всегда вопрос выбора.

Мне кажется, мы в очень многих вопросах, как в плане ведения бизнеса, так и в аспекте создания ПО – да и, по сути, любого продукта, – утратили понимание важности эстетической составляющей. Уоллес Стивенс когда-то написал сборник эссе «Necessary Angel» (Необходимый ангел) (Vintage). Одна из мыслей, которую он пытался донести, – что выбор всегда имеет значение. В «Записках всевышнему вымыслу» он развивает мысль о том, что Бог, возможно, является художественным вымыслом, фантастикой, в которую могут верить все. Цель религии, да и науки – создание художественного видения, концепции, в которую сможет уверовать каждый. И при ее помощи вы пытаетесь завербовать людей в свои ряды.

Эндрю: **Прямо вижу, как читатели, трудящиеся в офисах над созданием приложений для баз данных, читают это и думают: «В том, что я делаю, нет никакой эстетики».**

Тим: А я бы им сказал: «Посмотрите на Стива Джобса». Вот мой ответ. Для меня он живое доказательство могущества эстетики. Он принял ряд решений. Многие считают, что компания Apple терпит неудачи, потому что отказывается вписываться в господствующую парадигму. Он идет своим путем и следует собственному пониманию прекрасного. Ему удается возвращаться снова и снова, потому что он обладает неотразимой силой воображения, на которую люди неизменно покупаются. Ее называют «полем искаженной реальности Стива Джобса» – и этим все сказано. Он может создать притягательную и убедительную иллюзию, в которую поверят другие.

Помню, как одна из моих ключевых сотрудниц смеялась надо мной: «После разговора с Вами я ушла, согласившись со всем, что Вы говорили. И только потом вспомнила, что я во все это не верю!» У меня получалось ее убедить. Тут я хочу вернуться к наблюдению, которое может оказаться любопытным. Я помню, как численность моей компании перевалила за пятьдесят человек. Это был переломный момент, когда мне нужно было менять стиль работы, потому что первые годы работы в компании я действительно, так сказать, удерживал людей в своем «поле искаженной реальности» – всех сотрудников одновременно. Это была довольно небольшая группа людей, и мы работали в тесном контакте.

Помню, еще в школе по ночам я угонял машину своего отца. Чтобы он ничего не услышал, я толкал ее от гаража до дороги и еще целый квартал вниз, прежде чем завести двигатель. И вот толкал я эту машину и чувствовал только, что надо толкать, и постепенно она начинала двигаться

все быстрее. То же чувство у меня возникло и в отношении компании: «Ух, кажется, это слишком тяжело». Тогда я просто отпустил поводья. Старался общаться с не слишком большими группами людей. Даешь людям свободу делать свое дело, а потом просто проверяешь сделанное.

На заре своей карьеры я прочитал один научно-фантастический роман, который оказал на меня большое влияние. Это была книга «Рисса Кергелен» Ф. М. Басби (1977). Не самая известная книга. Одной из ключевых концепций, довольно тогда популярной, была концепция растяжения времени. При достижении скоростей, близких к скорости света, согласно одному из парадоксов Эйнштейна меняется инерциальная система, и время для человека, движущегося на такой высокой скорости, идет гораздо медленнее. И так во многих фантастических романах: люди улетают, возвращаются, а все вокруг сильно постарели. Так вот, «Рисса Кергелен» – это трилогия, и одна из частей называется «Дальний прицел». В ней говорится о том, как планировать космическое путешествие, если вернуться предстоит через 15 лет по планетарному времени. Идея такая: вы запускаете что-то в действие, а потом смотрите, что из этого вышло. Мне кажется, это очень мощный образ. И по сути то, о чем мы говорили с вами в начале, – архитектура различных систем – это способ запустить что-то в действие. Вы можете вернуться к этому и посмотреть, развивается ли оно так, как вы себе это представляли. Это навело меня на размышления о том, как нужно работать в компании. Нужно запускать что-то в действие, а потом возвращаться и смотреть, что из этого получилось.

Дженни: **И как реагировала ваша команда, когда вы начали таким вот образом отстраняться?**

Тим: Несомненно, у людей бывают периоды, когда они испытывают большой прилив сил. Они хотят вести других и менять направление движения. И мне кажется, один из признаков хорошего руководителя – умение распознать в ком-то из людей такие силы. Я фанат Celtics[1], и помню одну известную историю времен Ларри Бёрда. Идет решающий период баскетбольного матча, а тренером тогда был Кей Си Джонс. Команда собирается вокруг тренера, и Ларри говорит: «Просто пасуйте мяч мне и не мешайтесь». Кей Си отвечает: «Заткнись, Ларри, здесь я тренер! Все поняли? Пасуем Ларри мяч и не мешаемся».

То есть кто-то идет вперед, и ваша задача как тренера или руководителя сказать: «Он прав, дайте ему дорогу». У меня такие отношения сложились с Дэйлом Догерти. Вообще, O'Reilly во многом обязана своими успехами Дэйлу. В очень многих проектах, приписываемых мне, он сыг-

[1] Boston Celtics – профессиональный баскетбольный клуб, выступающий в Национальной баскетбольной ассоциации (НБА). – *Прим. перев.*

рал ключевую роль. Именно благодаря ему мы оказались во Всемирной паутине. Этот парень придумал название «Web 2.0». Сейчас он издает журнал «MAKE». Это похоже на танец, где партнеры расходятся, выполняют свои фигуры, а потом снова сходятся. Порой мы как братья, когда каждый рвется порулить. Иногда у нас бывает полная гармония. Особенно ценны люди, которые будут с вами спорить. Люди, у которых есть собственное видение.

Когда у вас есть представление о чем-то истинном – нет, я не знаю, что такое истина, абсолютная она или художественная – когда некоторые вещи просто работают, когда вы слышите идеальный аккорд, видите линию рисунка или безупречный изгиб каменного изваяния, на ум сами собой приходят знаменитые строки Китса: «В прекрасном – правда, в правде – красота. Вот знания земного смысл и суть»[1], – именно так все собирается воедино. Для меня ключ к объединению людей для совместной работы – ваше художественное видение, концепция, которую вы сможете разделить с другими. Когда вам удается создать общее понимание истины, к которой вы идете, когда у вас получается выразить свое понимание идеала. Потому что тогда вы даете им свободу и пространство, чтобы двигаться к этому идеалу самостоятельно.

[1] Джон Китс «Ода к греческой вазе». Пер. В. Микушевича. – *Прим. перев.*

I

Люди

Когда мы начали работать над книгой «Идеальные команды» в конце 2007 года, нам казалось, что мы довольно четко представляем, о чем собираемся писать. Мы «технари» – внедрили немало методов, запустили большое количество проектов разработки и с ходом времени наблюдали, как дела идут в гору благодаря нашей работе.

Мы думали – поговорим с людьми, которые думают так же, как мы, законспектируем их интуитивные прозрения, секреты мастерства и мудрые советы, и все будут счастливы.

Оказалось, нас ждало немало сюрпризов.

Надо сказать, мы не были полностью неправы, и направление было выбрано верно. Результаты говорят сами за себя, и это хорошие результаты. Одной из причин нашей уверенности в том, что мы довольно хорошо разобрались, как работают хорошие команды, стали мнения многих других людей, которые согласились с нами.

В чем мы оба сходились, начиная этот проект, так это в том, что в хорошей команде главное – техническое мастерство ее участников. Поэтому когда мы начали привлекать авторов и читать их материалы, мы ожидали услышать немало рассказов о том, как обеспечить техническую состоятельность своей команды: искать и нанимать нужных людей, применять нужные методы (скажем, при написании технических требований или оценке кода) и обучать им сотрудников, в общем поддерживать и развивать тех, кто проявил себя талантливым и многообещающим программистом.

Все обернулось несколько иначе. На деле, во всей книге совсем немного рассказов о том, как обучать людей, совершенствовать их навыки или придирчиво отбирать людей в команду.

Тут нужно уточнить: никто не говорил, что это не важно. Даже наоборот, сложилось впечатление, что все наши авторы исходили из предположения о том, что хорошая команда начинается как минимум с формирования ядра хороших специалистов с требуемым образованием, опытом работы и отношением к делу. Для всех было очевидно, что если сотрудники не справляются с работой, то и команда, и проект обречены. Открытием для нас стало то, что набор нужных специалистов в команду оказался не конечной целью, но отправной точкой для развития проекта.

Истинное положение вещей оказалось куда сложнее, чем мы полагали поначалу. Определяющие его факторы с трудом поддаются подсчету и определению. Порой лучшие члены команды вовсе не являются самыми крутыми специалистами, а самые технически сильные сотрудники могут только мешать выполнению работы. Люди должны уметь работать вместе, и если они этого не могут, технические навыки уже

ничего не решают. На деле, некоторые самые технически продвинутые программисты рискуют слишком зазнаться и могут, по сути, осложнить реализацию проекта. Но чаще всего именно эти люди наиболее важны для команды.

Мы заволновались. Мы привыкли к четкой полярности мира: либо у вас команда лучших специалистов (самых умных, технически подкованных, талантливых разработчиков, тестировщиков, архитекторов и т. д.), которые могут выполнить работу идеально, либо ваша команда не так хороша, и с этим нужно что-то делать. Но мы обнаружили, что *все* команды неидеальны. Даже в коллективах из блестящих высококвалифицированных специалистов могут возникать серьезные проблемы, которые отразятся на качестве программирования. В *каждой* команде есть место эмоциональной составляющей. Даже те, кто создает безупречный, исключительно надежный продукт, неизбежно сталкиваются с обычными сложностями, обусловленными самим характером человеческого взаимодействия.

Осознав это, мы обратились к собственному опыту. И обнаружили, что самые успешные команды не всегда состояли только из самых продвинутых программистов. Лучшими были те команды, члены которых работали наиболее слаженно.

В конце концов, команды состоят из людей, и главным слагаемым в этом уравнении являются люди. Иногда и мы впадали в обычное заблуждение о том, что специалисты с равной квалификацией могут быть взаимозаменяемы. Если вы считаете разработку ПО исключительно техническим процессом, очень легко начать рассматривать людей как ресурс, распределяемый на выполнение заданий, а их работу оценивать в человеко-часах; нетрудно прийти к заключению о том, что хорошие люди – лишь винтики в механизме, а их квалификация – качество, подлежащее измерению и улучшению при помощи сертифицирования, обучения и опыта работы.

Наши авторы и собеседники не раз показали нам, насколько ошибочно такое мнение. И мы вдруг поняли, что и у нас были случаи, когда мы начинали воспринимать людей как взаимозаменяемые шестеренки общего механизма, и что это неизбежно приводило наши проекты к серьезным проблемам.

В этой части книги собраны как раз те рассказы и интервью, которые напомнили нам об этом важном уроке. Это истории о том, к чему приводят конфликты в командах, и о том, что может получиться, если людям удается работать слаженно. Читая эти рассказы и беседуя с этими людьми, мы поняли, что не существует образцовых команд блестящих специалистов, идеально совместимых и профессионально работающих на протяжении всего срока проекта.

Более того, самые важные для команды сотрудники далеко не всегда лучшие в программировании. Это даже не руководители или лидеры по натуре. Это те, кто умеет быть *настоящим*, – они просто делают свое дело и остаются собой в любой ситуации. Они прислушиваются к окружающим и понимают, что их мотивирует и что раздражает. Совсем не потому, что они прочли специальное пособие или прошли курс мотивационного лидерства. А потому, что они умеют быть внимательными к людям. Они не жалеют времени на то, чтобы все в коллективе научились общаться на одной частоте, причем не обязательно на уровне взаимной любезности или даже взаимопомощи, а на том уровне совместимости, который для них приемлем и действенен. Такие люди чувствуют ситуации, в которых совместная работа невозможна (вне зависимости от причины), и пытаются их исправить.

В командах все очень непросто. Они пронизаны эмоциональными связями между людьми, работающими на пределе умственного напряжения над задачами, которые могут оказаться неразрешимыми. И когда в такой ситуации у вас есть те, кто посреди этого хаоса остаются собой, готовы идти в огонь, прислушиваться к коллегам, вместе работать допоздна и преодолевать разочарования, то вы получаете отличную команду. Не обязательно *идеальную*, но эффективную и комфортную для работы. Без таких взаимоотношений между людьми в коллективе работа остается лишь работой, а команда никогда не станет тем коллективом, о котором его сотрудники будут вспоминать до конца своей жизни.

2

Почему побеждают ужасные команды

Скотт Беркун

«Несносные медведи». Ramones. Рокки Бальбоа. «Грязная дюжина».[1] Настоящие герои ужасны. Они всегда и везде не к месту. Они не так одеваются, не так себя ведут и понятия не имеют, как надо. Над ними смеются, им в открытую говорят, мол, ради всего святого, брось ты эту затею, но они не слушают. Вопреки неудачам, они идут к своей цели, полагаясь только на свою настойчивость, увлеченность и находчивость. И именно по этой причине, когда наступают сложные времена, такие «гадкие утята» выходят вперед. Эти люди заранее готовы к тому, что все пойдет не так, и сознательно идут на это. Они преодолевают неуверенность, находят друзей на передовой и видят волшебство в том, что другие выбрасывают за ненужностью. Эти слаженные рабочие механизмы не горят в огне и в воде не тонут, и, завоевав доверие друг друга, дадут фору любым другим коллективам любой компании. Слова Ницше идеально подходят для ужасных команд: то, что не убивает такую команду, делает ее сильнее.

Ужасные таланты

Многие так называемые идеальные команды не всегда были таковыми в представлении их членов. Лу Гериг и Бейб Рут – игроки, возможно,

---

[1] «Несносные медведи» (The Bad News Bears) – комедия Майкла Риччи (1976 г.) и одноименный римейк Ричарда Линклейтера (2005 г.) о детской бейсбольной команде. Ramones («Рамоунс») – одна из первых американских панк-рок групп. Рокки Бальбоа (Rocky Balboa) – боксер, герой одноименной драмы Сильвестра Сталлоне (2006 г.). «Грязная дюжина» (Dirty Dozen) – военный боевик Роберта Олдрича (1967 г.) – *Прим. перев.*

величайшей спортивной команды в истории, Нью-Йорк Янкиз[1] времен 1927 года, – презирали друг друга. Отцы-основатели США Томас Джефферсон и Джон Адамс постоянно враждовали, на людях и в личном общении. Многие великие музыкальные группы, к примеру, The Supremes, The Doors, The Clash, The Beatles и даже Guns N' Roses, смогли продержаться вместе лишь несколько лет.[2] Нам нравится простая мысль о том, что только красивый человек или красивая команда может создать что-то прекрасное. Как будто Пикассо не был социопатом и женоненавистником, Ван Гог – маниакально-депрессивным психопатом, а Джексон Поллок –алкоголиком (и десятки других известных творческих людей и легендарных спортсменов – наркоманами). Значение красоты завышено, и многие из работы не считаются прекрасными долгое время после их создания или после смерти автора (но если работа остается той же, значит меняется что-то другое – что же?). Большинству из нас свойственна искаженная, неестественная, чрезмерно упрощенная эстетика, согласно которой все, что красиво, хорошо, а все, что некрасиво, плохо, без вариантов.

В бестселлере 2003 г. «Moneyball: The Art of Winning an Unfair Game» (Деньгобол: искусство победы в нечестной игре) Майкла Льюиса (W. W. Norton & Company) описываются критерии, по которым агенты бейсбольного клуба «Окленд атлетикс» оценивали способности новых игроков. Вместо того чтобы смотреть только на результаты, на умение игрока бить по мячу или пасовать так, чтобы отбить мяч было невозможно, профессиональные агенты в первую очередь смотрели на внешность. Игроки с лишним весом, маленьким ростом или плохой (на первый взгляд) координацией отсеивались, невзирая на статистику, подтверждающую их мастерство. Билли Бин, менеджер «Окленд атлетикс», коренным образом изменил систему оценки потенциала игроков, устранив это противоречие между нашим представлением о том, как должен выглядеть талантливый игрок, и тем, как он выглядит на самом деле. Он заставил людей отказаться от предвзятости и подходить к оценке таланта менее субъективно. Некрасивые игроки или красивые, но не слишком «зрелищные» в игре, могли показывать хорошие результаты и были гораздо ценнее, чем считалось в лиге. Его честный взгляд на то,

---

[1] «Нью-Йорк Янкиз» (New York Yankees) – профессиональный бейсбольный клуб, выступающий в MBL (Major League Baseball). Самая успешная команда MBL, выигравшая 27 игр Мировой серии – *Прим. перев.*

[2] Пока не пройдет лет десять, и возможность заработать не перевесит их взаимную неприязнь. См. *http://www.spinner.com/2007/08/10/20-bitter-band-break-ups-smashing-pumpkins/*, где приведен более длинный список знаменитых распавшихся групп.

что для бейсболиста по-настоящему важно, изменил подход к оценке и подбору игроков во многих профессиональных командах.

Подобно бейсбольным агентам до появления Бина, у всех нас есть четкие представления о людях, которые нам сложно обосновать. В течение всей жизни у нас подспудно складывается образ того, как должен одеваться, разговаривать и вести себя великий спортсмен, внушающий доверие доктор или блестящий программист, и этот образ влияет на наше мнение больше, чем мы осознаем. Когда мы думаем о командах, большинство наших представлений о том, как должна выглядеть хорошая команда, почерпнуты из телепрограмм и кино.[1] Немногие герои и легендарные команды были в реальной жизни столь же неотразимы и круты, как сыгравшие их звезды, и мало у кого из них история развивалась так гладко и быстро, чтобы ее можно было изложить в 90-минутной картине. В таких фильмах, как «Самородок», «Спасти рядового Райана» или даже «Матрица», опускаются все неприятные стычки и борьба, в которых рождается и развивается команда, и нам показывают только историю о том, как в команды красивых успешных людей приходит красивый одаренный лидер.

Вопрос на засыпку: перед вами два кандидата на должность, один – просто чудо с идеальным средним баллом в аттестате (4), а второй – возможно, очень умный, но с «выборочной мотивацией» и средним баллом в 2,7 (два высших балла и 4 оценки ниже среднего)[2]. Кого вы возьмете? При равных прочих мы все выбрали бы безупречного. За такого работника никто не уволит. Что может быть лучше, красивее идеальных оценок? Однако, если копнуть глубже, идеальные оценки часто означают идеальное умение следовать чьим-то правилам. Они не свидетельствуют об увлеченном, свободном, бесстрашном уме. Что еще важнее, команда, состоящая только из гениев с идеальным средним баллом, никогда не станет ужасной. Они не пойдут на большой риск, не совершат больших ошибок и, соответственно, им никогда не придется спасать друг друга из беды. Но без этих рисков, ошибок и взаимовыручки невозможно создать связи, основанные на глубоком личном доверии. Чтобы команда могла сотворить что-то прекрасное, ее путь не должен быть безупречен. Трагедия команд, состоящих из безупречных людей, в том, что они будут так отчаянно стараться сохранить свою безупречность, свои высшие баллы, что когда они столкнутся с прессингом серьезного проекта, их честолюбие разрушит команду. Красивые люди слишком боятся шрамов: у них не достанет воображения увидеть красоту этих шрамов.

---

[1] Да, я помню, что упоминал здесь «Несносных медведей» и «Рокки Бальбоа». Даже у лучших выводов бывают исключения.

[2] Признание: средний балл автора, кажется, тоже около того.

Когда ужасное прекрасно

Прекрасный и *ужасный* – неоднозначные определения для групп людей. Когда я называю прекрасными Мону Лизу или пик Мак-Кинли, я говорю о том, что они приятны глазу или хорошо исполнены: я даю эстетическую оценку объекта, который мы можем наблюдать вместе. Я могу указать на него, описать его, забросать его помидорами или даже позволить вам сравнить ваше суждение об этой вещи, какой вы ее видите, с тем, как я описываю ее, увиденную мной. Я верю в то, что красота в глазах смотрящего, но в данном случае двое людей с разными взглядами все же говорят об объекте, существующем вне каждого из них. Но в отношении команды, клуба или даже страны слово «прекрасный» менее понятно. Команда определяется отношениями между людьми, а отношения не предметны. Суждение об эстетике нематериального значительно расширяет пределы самого понятия эстетики. Оно переносит красоту из глаз смотрящего в его разум, где уже невозможно смотреть на предмет вместе с ним. Руперт, капитан команды, думает о команде одно, а Корнелиус, талисман команды, другое. И, разумеется, люди из сборной противника имеют о ней свое, отличное мнение. И ни один из них не может указать на «команду» как на объект и единую точку отсчета с той же уверенностью, как они могли бы указать на Мону Лизу или пик МакКинли.

Единственное имеющее смысл понимание красоты, применимое к командам, это японская концепция *ваби-саби*. Грубо говоря, *ваби-саби* означает особую красоту вещей, которыми уже пользовались. Та пара ботинок, которую вы любите за то, что они хорошо разношены и несли ваши ноги в долгих прогулках вдоль берега, обладает красотой, какой не будет ни у одних новых ботинок. Даже грязными и потертыми, такими, что ни один человек, присматривающий себе новую пару ботинок, не смог бы назвать их красивыми, они по-прежнему будут обладать для вас красотой в смысле *ваби-саби*.

Иногда бывает, что некоторые вещи в своем старении становятся прекрасными для всех. Посмотрите на самое старое здание в вашем районе, самое старое дерево в вашем парке. Есть особое величие в том, как старятся некоторые вещи, и оно напрямую связано с теми изъянами и несовершенствами, которые эта вещь приобрела с течением времени.[1]

[1] В ходе последней реставрации Пантеона в Греции рассматривалась идея воссоздания строения в его первоначальном виде. Но в итоге было решено реставрировать его в те же руины, что и сейчас, ибо эстетика обработанного погодой и временем камня и потертого мрамора лучше соответствует представлениям о том, как должно выглядеть это здание. Принцип *ваби-саби* восторжествовал.

Любой, кто предпочитает покупать старые вещи, в том числе и из-за их вида, понимает, что такое *ваби-саби*. В этом смысле, описанные в начале этой главы ужасные команды аутсайдеров, неудачников можно назвать командами *ваби-саби*. Это люди, у которых общие шрамы, они падали и поднимались вместе и продолжают драться как единая команда. И только в результате таких испытаний команда может выработать тот характер, который в каком-то приближении можно назвать красотой.

Моя команда ваби-саби: Internet Explorer 4.0

В 1995 году я начал работать в Microsoft, в команде Internet Explorer. Это был небольшой молодой проект, над которым работала горстка людей. Он даже не вошел в Windows 95. Первый веб-броузер Microsoft был выпущен отдельно, в виде расширения для Windows, стоившего 49 долларов и известного как Plus Pack.[1] Но с подъемом Netscape и надеждой всей отрасли на то, что этот подъем возвестит о конце Microsoft, значение нашей команды резко возросло. Руководство Microsoft, всегда страдавшее паранойей и съевшее собаку на погонях за чужим успехом, как известно, тут же запустило свою кампанию, сделав Интернет стержнем всей стратегии и тактики компании. К версии 4.0 команда проекта состояла более чем из сотни человек и занимала два этажа в здании 27 на севере городка Microsoft.

В 1997 году команда Internet Explorer начала судьбоносную разработку версии 4.0. За всю историю разработки ПО немногие проекты сталкивались с таким количеством проблем, как мы в течение всего лишь года.[2] Бесконечные реорганизации, битвы в высших эшелонах, утечки планов разработки, невыполнимые задачи, антимонопольные иски в Минюст, бесконечная смена руководителей среднего звена, и все это под грузом ответственности за спасение компании от величайшей угрозы (по крайней мере, по мнению остальных компаний в отрасли) в ее истории.[3] Если бы нам подбросили еще пару эпидемий и стихийных

---

[1] Даже наши ребята из маркетинга не были уверены, что этот броузер «выстрелит», в то время более надежные вещи вроде Рабочего стола, компрессора дисков (ну да, это же 1995 год) и планировщика фоновых задач приносили такой же или больший доход.

[2] Не то чтобы я гордился этим, но я еще не встречал истории более драматичной, чем у проекта IE4.0, учитывая сцену, на которой она разыгрывалась. Если у вас есть такая история, я с удовольствием ее выслушаю. Мы, уцелевшие в сложных проектах, всегда рады компании.

[3] Необязательно верить мне на слово. Об этом периоде в Microsoft написано две книги. Поищите «How the Web Was Won» (Как выиграли Web) Пола Эндрюса (Broadway), до смешного положительную в отношении всего, что

бедствий, мы могли бы поставить галочки против всех пунктов в списке катастроф, которых должен избегать любой менеджер.

Но ловушку, которая могла привести к гибели команды, мы поставили сами: отчаяние и высокомерие. Первые три версии были успешными. Internet Explorer 1.0 был простой модификацией купленного у Spyglass броузера. Версия 2.0 разрабатывалась очень быстро и была выпущена уже через несколько месяцев. Затем 3.0, первый крупный выпуск, показал миру, что Microsoft еще жива, догнал остальных, добавил новых идей и был вполне конкурентоспособен. И когда у обеих сторон был накоплен арсенал ресурсов, началась четвертая волна войн за броузер. Обе стороны сделали самые крупные ставки и были не в состоянии понять, что характер проекта изменился коренным образом. Как самонадеянный жонглер, внезапно осознавший, что он подбросил больше шариков, чем может сосчитать, не говоря уже о том, чтобы поймать, наша команда мгновенно посыпалась.

Наша главная головная боль называлась «Каналы». В 1997 году весь мир был убежден, что будущее Сети за push-технологией, возможностью веб-сайтов «проталкивать» информацию посетителям (предшественницей RSS-каналов, которые сегодня используются в блогах). Не люди должны были искать информацию в Сети, а контент находить путь к людям, автоматически загружаясь и появляясь в их закладках, на рабочих столах, в электронной почте или в виджетах, панельках рабочего стола, которые еще только предстояло разработать. Мы называли это «Каналами», и ими занималась небольшая команда. Пока они бились над разработкой этой функции, у ребят из коммерческих отделов шла гонка с конкурентами из Netscape за то, чтобы заполучить себе ключевые веб-сайты вроде Disney или ESPN.[1] Нам нужен был их контент, чтобы все могло заработать: одно дело иметь канал – нужна еще и информация для «проталкивания» через этот канал.

Мы потерялись во всей этой чехарде безумных сделок и попыток угнаться за общей шумихой. Невозможно стать новатором, держа одну руку на своде правил и сунув другую в огонь. В результате заключаемых сделок на руках у разработчиков оказывались договоры и юридические обязательства: использование данных с этих веб-страниц имело мно-

---

связано с Microsoft. В отличие от нее, в книге Майкла А. Кузумано и Дэвида Б. Йоффи «Competing on Internet Time» (Конкуренция за эпоху Интернет) (Free Press) изложена более уравновешенная версия с точки зрения Netscape, касающаяся, в основном, стратегии, а не личностей и тактических приемов.

[1] ESPN – крупнейший американский спортивный интернет-ресурс. – *Прим. перев.*

жество ограничений, и мы должны были их соблюсти, несмотря на то что немногие из разработчиков видели эти договоры и обязательства до их подписания. Как у «Титаника», вышедшего в океан с тысячей бракованных заклепок, наша судьба была предрешена задолго до начала скандала. Несмотря на месяцы работы, команда «Каналов» не смогла разработать продукт. Демоверсии были очень путаными, ответы на базовые вопросы – еще хуже. Вскоре молва о «Каналах» распространилась на всю команду и компанию и потянула вниз за собой репутацию всего проекта. Если бы это было главной ставкой для Microsoft, мы бы уже проиграли.

Команда Internet Explorer всегда была ареной для высказывания различных мнений – громких, запальчивых, саркастичных, а порой и оскорбительных. Разногласия среди руководства привели к остановке и бездействию в работе, и те, кому было что сказать, усилили громкость и язвительность. В этом горниле страстей любой был обречен, в итоге менеджер проекта по «Каналам» спекся. Вскоре сменили и его, и его начальника, а затем и этих новых людей. В этой неразберихе, никем не контролируемые, изворотливые щупальца «Каналов» стали расползаться по проекту, поражая код, системы и общий боевой дух. Если что-то не так с крупными проектами, некомпетентным оказывается каждый. Все становятся ужасными. Люди начали уходить. Росло отчаяние. Менеджеры отрывались на совещаниях, по переговорным летали тяжелые предметы. Шли месяцы и росли счета от психотерапевтов. Завершая другие элементы проекта, мы старались не замечать черную дыру в виде «Каналов» в самом центре проекта, постепенно засасывающую все вокруг.

Не знаю, когда это началось, но где-то после четвертой реорганизации, при третьем генеральном менеджере и пятом менеджере проекта «Каналов» начался период висельного юмора. Именно в такие моменты и зарождается основа будущей команды *ваби-саби*. Не оправдав никаких ожиданий, мы переключились в тональность черного Беккетовского юмора. Это началось, когда нас поставили перед очередным нелепым, идиотским, самоубийственным выбором, когда все варианты решения были равно плохими. «Почувствуй любовь», – сказал кто-то. Это была дурацкая фраза из серии «помоги себе сам», но это было так далеко от окружающей нас действительности, что это сработало. Иногда в письмах, делая заведомо нелепый запрос, мы добавляли после этих слов смайлик. Или мы хлопали друг друга по плечу с этими словами, подчеркивая фальшивость и шаблонность такого поведения.

Это работало, потому что мы знали, что все мы в одной дырявой лодке, просто в тот конкретный день на долю одного из нас выпало чуть больше. Однажды на моих глазах месяцы работы команды, включая

мою, бездумно выкинули, просто вычеркнули наряду с другими пунктами, без всякой логики и без возможности защититься, и кто-то написал в письме: «Почувствуй любовь! Это IE4!» Позже эта надпись ждала на доске в комнате, где собрались руководители группы. Даже менеджер группы не мог не рассмеяться, когда увидел ее, объединяясь с нами в общем сардоническом настрое. В тот момент и для меня, и для группы что-то изменилось: он был с нами на одной волне. Если мы не могли уйти от судьбы, по крайней мере мы могли честно называть вещи своими именами.

В конце концов я стал шестым – и последним – руководителем программы «Каналы». Моей задачей было быстро подготовить что-то для окончательного бета-выпуска и устранить все неполадки, какие успею, прежде чем программа будет окончательно запущена. Когда мы выпустили ее, получив преимущественно положительные отзывы, у нас состоялось самое безумное торжество в моей жизни. Дело было не в шампанском, не в месте проведения и даже не в количестве гостей. А в том, как мало всего было съедено за многочисленными столами: в первые же минуты большинство блюд было с любовью запущено в коллег и руководителей. Я стал мишенью для самого большого комка гуакамоле[1], когда либо прилетавшего в голову человеку (где-то даже должна быть фотография, подтверждающая это).

Настоящие связи *ваби-саби* между нами укрепились после этого. Те немногие, кто остался работать над Internet Explorer 5.0, были связаны уже особым образом. Мы видели друг друга в худшие моменты, и все же чувствовали уважение друг к другу. Мы все знали, как ужасно то, что может случиться, и могли положиться друг на друга в том, чтобы не допустить повторения этого. На одном из первых совещаний по планированию весь разговор крутился вокруг того, как убить «Каналы», стерев их с лица проекта. В последующие месяцы мои возможности и полномочия как руководителя подкреплялись тем, что я мог прямо смотреть в глаза некоторым программистам и полностью им доверять, поскольку лично видел, как они справлялись со сложными ситуациями, то же они чувствовали по отношению ко мне. Между нами было доверие, родившееся в некрасивых, отчаянных, но совместных боях, и мы могли фокусироваться на реальных проблемах пользователей независимо от общей шумихи и тенденций, прорицаемых знатоками. Internet Explorer 5.0 был лучшей проектной группой, где мне довелось работать, и одним из лучших релизов ПО в истории Microsoft. Может, это мало что значит для других, но для меня эта команда прекрасна.

[1] Гуакамоле – мексиканское блюдо, соус из мякоти авокадо. – *Прим. перев.*

3

Создание видеоигр

Интервью с Марком Хили

LittleBigPlanet («Маленькая Большая Планета») была одной из самых ожидаемых игр 2008 года на Play Station 3 и по разным оценкам стала одной из лучших игр года. Это не только жутко увлекательный платформер, но и творческий инструмент, позволяющий игрокам строить собственные уровни и делиться ими друг с другом. Комбинация отличного игрового процесса и социальной сети стала победной для создателей игры, компании Media Molecule, основанной несколько лет назад Марком Хили. Нам стало любопытно, каково это, работать над такой игрой, и вот мы у Марка.

Эндрю: **Дженни просто с головой ушла в LittleBigPlanet, и мне это нравится. Вы сделали потрясающую игру.**

Марк: Спасибо большое. Если честно, я сам был сильно ею увлечен.

Дженни: **Это великая игра! Правда. Поэтому нам и захотелось поговорить с вами. Услышать историю создания игры. И, поскольку наша книга на самом деле о командах и о том, как они складываются, было бы здорово услышать, как вы собрались вместе для ее создания.**

Марк: Дайте подумать. Придется вспоминать, а с этим у меня обычно туго...

Компанию основали Алекс Эванс, Дэйв Смит и Карим Эттуни при большой поддержке Криса Ли, очень хорошего администратора.

Все мы, кроме Криса, прежде работали вместе в компании Lionhead Studios и дружили.

Мы постоянно делали что-то вместе, и однажды я придумал снять на выходных простенький фильм о кунг-фу, так, смеха ради. Взял в Lionhead камеру, и фунтов 50 мы потратили на костюмы и реквизит. Пошли в парк у меня за домом и сняли исключительно глупый фильм про кунг-фу, без всякого сценария и плана; мы понятия не имели, кто придет, поэтому просто сочиняли на ходу. Потом я начал монтировать отснятое в маленький фильм – и параллельно учиться программировать на C++, потому что вообще-то давным-давно я был программистом, еще до того как стал художником. В те времена я писал на Assembler (6510), но решил податься в искусство, потому что с ним меньше мороки.

И вот я монтировал фильм и изучал C++, а потом сложил два и два и подумал: «Хорошо. Раз уж я изучаю C++, нужно придумать небольшой проект. Напишу-ка я простую боевую игру, что-то в духе Street Fighter, а из фильма про кунг-фу сделаю для нее видеовставки». Так появился инди-проект, небольшая игра Rag Doll Kung Fu (рис. 3.1), в конечном итоге ставшая первой игрой от независимых создателей, выпущенной Valve в системе цифрового распространения Steam. В те времена компании Media Molecule не было еще даже в проекте.

Работая над этим проектом в свободное время, я нередко по-дружески просил Алекса и Дэйва помочь мне с какими-нибудь сложными техническими вещами. Программировать – одно дело, но программировать на PC – сплошная головная боль. Лично для меня это невероятно сложно, читать документацию по DirectX, просто ужас какой-то, там что ни слово – ссылка на очередную страницу! Мне очень повезло, что Алекс и Дэйв такие умные и технически продвинутые, они за 10 минут сделают то, на что у меня ушло бы 10 лет...

Рис. 3.1. *Кадр из игры Rag Doll Kung Fu, впервые собравшей эту команду вместе*

С Каримом я познакомился за пару дней до того, как позвал его поучаствовать в нашем кунг-фу в парке, вскоре мы стали лучшими друзьями, в результате чего он был весьма активно задействован в Rag Doll Kung Fu – сыграл злого главаря в фильме, помог кое-что отрисовать, вместе мы сделали музыку к игре на четырехголосном синтезаторе у меня в гостиной; мы даже записали озвучку смешными голосами с ним и другим моим другом Барри, а его девушка Сиобан позже стала незаменимым исполнительным продюсером в команде Media Molecule.

Как видите, мы спонтанно собрались в команду на проекте Rag Doll Kung Fu. И среди инди-проектов наш был вполне успешным. Для такой небольшой игры он привлек большое внимание и, что важнее, дал нам почувствовать вкус к чему-то большему. Мы всегда работали на кого-то, и создание небольшой независимой игры стало для нас как бы экспресс-курсом по всему процессу: от разработки и производства до локализации и выпуска…

Эндрю: **То есть для вас все началось с проекта Rag Doll Kung Fu?**

Марк: Rag Doll Kung Fu собрал нас вместе. Потом через общих знакомых мы вышли на Криса, у которого был опыт руководства компанией (он руководил Renderware), и он указал нам, над чем нужно подумать и что нужно сделать. И так вышло, что в самое ближайшее время через общего друга нам выпала возможность выйти со своим проектом на

Фила Харрисона – тогда он возглавлял Sony – но штука в том, что у нас не было времени подготовиться! Встреча была назначена через неделю. Нам нечего было показывать, поэтому пришлось сочинить что-то максимально быстро (рис. 3.2).

Рис. 3.2. *Марк в видеозаставке из Rag Doll Kung Fu*

Эндрю: **Да, работали вы, наверное, лихорадочно – причем во всех смыслах.**

Марк: Это было страшно интересно и, наверное, напряжение было очень большим. Но, знаете, и прежде это всегда было нам только на пользу, такие жесткие временные рамки.

Дженни: **Именно так вы и сблизились друг с другом, когда еще работали вместе в Lionhead?**

Марк: Ну да. То есть ближе к концу нашей работы в Lionhead люди вроде меня, Алекса, Карима и Дэйва часто собирались вместе, чтобы решать какие-то проблемы, но когда не было таких экстренных вопросов, мы были вольны делать что хотим, правда. Нас сажали и просили придумать что-нибудь классное. Кажется, это называется НИОКР.

Эндрю: **Вам комфортно в такой обстановке? Многих пугает отсутствие рабочих указаний, без которых они не могут работать.**

Марк: Лично мне это очень нравится, но тут, конечно, требуется умение мотивировать себя. Нужно уметь составить хотя бы грубый план

или идею того, куда вы будете двигаться, иначе очень легко так и просидеть, ничего не делая или бездумно лазая по Интернету. Мы работали над проектом «The room» (Комната). Думаю, где-нибудь в Сети можно найти это видео.

Дженни: **Кроме вас троих — сколько еще людей работает в Media Molecule?**

Марк: На сегодня нас около 30 человек.

Дженни: **Все эти люди работали и над первой версией LittleBigPlanet? Каким образом вы планировали этот проект?**

Марк: Когда мы поняли, что будем представлять проект Филу Харрисону, мы в общих словах обговорили, чем все мы хотели бы заниматься. На самом деле, идеи были из разряда недостижимых и не очень хорошо сформулированы, поэтому нашей основной стратегией было подать себя как невероятно изобретательных, творческих и умелых людей, которые при финансовой поддержке Sony непременно создадут что-то потрясающее...

У Дэйва, например, был большой опыт работы с физикой, он разработал очень крутой физический движок 2D и мы точно хотели его где-нибудь применить. По поводу игры Rag Doll Kung Fu, хотя это и был маленький независимый проект, мы знали, что многое в нем очень понравилось пользователям, и в особенности возможность создавать собственные персонажи, запускать их в игру и отыгрывать ими. То есть у нас была идея насчет такой комбинации создаваемого пользователями наполнения и, если угодно, самовыражения.

Можно сказать, моим взносом была Rag Doll Kung Fu, Дэйв предложил физику, а Алекс очень хотел показать свое техническое мастерство — ему не было равных в программировании графики (он разработал графический движок). Также, по-моему, он больше, чем все мы, общался с детьми, и это он придумал использовать ресурсы YouTube и MySpace. Нам удалось собрать все это в небольшую, переносную презентацию и демоверсию, которая неожиданно оказалось очень крутой. Поищите в YouTube «Yellow Head» и увидите, как выглядела анимированная часть презентации. Все это мы собрали меньше чем за неделю благодаря коду, написанному Дэйвом в ninja physics.

Таким образом, эта встреча была демонстрацией и наших талантов, и очень конкретной идеи. Хвала небесам, Крис был вооружен магическим артефактом под названием «бизнес-план», который сообщал Sony, сколько нам потребуется денег и людей для того, чтобы разработать более полноразмерную демоверсию... Фил Харрисон был должным обра-

зом впечатлен всем этим, уступил нашим просьбам, и от этого все и закрутилось.

После встречи в Sony мы оперативно приняли Карима на должность арт-директора и начали смотреть, кто еще нужен нам в команду. Быстро нашлись и другие художники и программисты, большинство из которых были нашими друзьями (и очень талантливыми). Сиобан занялась организационными вопросами, а Мэгз (которая помогала мне в налоговых вопросах по доходам от Rag Doll Kung Fu) стала нашим бухгалтером и следила за тем, чтобы каждый получал деньги за работу и чтобы мы выплачивали все, что требуется от компании, принимающей на работу людей.

Процесс разработки представлял собой, в основном, открытое обсуждение, за которым следовало изложение вариантов на экране или на бумаге, или возможность «поиграть» с ними, или еще что-то. Моей задачей было собирать эти идеи в единый документ и пытаться увязать их, так чтобы представить всем результат. Таким образом, всем было понятно, чем мы занимаемся и куда движемся, а этот документ превращался в постоянно эволюционирующую проектную документацию.

Шесть месяцев пролетели быстро, и у нас была готова очень милая презентация для Sony (они каждый месяц проверяли результат, чтобы убедиться, что мы не пропиваем все деньги). Это довольно формальная и устрашающая встреча, получившая название «зеленый свет», и именно там принимается решение, перейдет ли проект в полномасштабное производство или его зарубят. К счастью, в Sony были очень довольны нами – настолько, что решили включить нашу демонстрацию в программную речь Фила на Конференции разработчиков компьютерных игр (КРИ), обращенную к более чем пятитысячной аудитории!

Дженни: **На мой взгляд, самое замечательное в вашей игре – это возможность создавать уровни и соответствующий пользовательский интерфейс. Как вы пришли к этому?**

Марк: На самом деле, это все пришло со временем. Когда мы в первый раз заявляли об игре, это было на уровне: «У нас тут двумерный платформер с физическим движком, и мы хотим добавить к нему возможность создания контента пользователями». Именно в таком виде мы предлагали это Sony. И их на самом деле в большей степени заинтересовал именно аспект пользовательского участия, что нас несколько удивило, потому что мы полагали, что их это напугает. Но как только мы поняли, что здесь они с нами заодно, мы активно начали работать над этим.

Одним из наших больших преимуществ стало то, что при первоначальном представлении проекта у нас была рабочая демоверсия, и мне

кажется, это сыграло большую роль в том, что они нами заинтересовались, потому что мы предлагали продукт, который можно было посмотреть и поиграть в него. По-моему, большинству издателей уже до смерти надоело смотреть презентации в PowerPoint.

Эндрю: Вы заговорили о том, как заинтересовать людей при первом представлении проекта. Получалось ли у вас поддерживать тот же уровень воодушевления, сначала среди вас, самих создателей, а потом и в расширенной команде? Движущий мотив у вас самих, должно быть, очень сильный. Повлияло ли это на процесс генерации новых идей? На то, как вы в целом разрабатывали программу?

Марк: Да. На самом деле, так мы всегда и работали – мы не пытались создать проектную документацию, четко прописывающую «Вот что мы будем делать». На первых этапах все очень расплывчато: «Ну, хорошо. Мы видим это как двумерный платформер с физическим движком и создаваемым пользователями контентом». Сам процесс работы вызывает к жизни новые идеи, и потом все развивается так: «Отлично. Это только что навело меня на мысль сделать вот так. Проверьте, получится ли». И мы действительно подхватываем идеи друг у друга. То есть мы работаем в атмосфере импровизации, джема, как у музыкантов, если вы понимаете, о чем я, что возможно только для небольшого количества людей. Как только вас становится больше, чем шесть или семь человек, вам приходится быть более организованными. Иначе все скатывается в хаос. Мы решили эту проблему таким образом: когда мы начали расширяться, мы создали «молекулы», небольшие группы людей, работающие в определенных направлениях. Это позволяет людям продолжать импровизировать.

Демонстрационная версия, представленная на Конференции разработчиков компьютерных игр, была практически идентична той, что мы представили на совещании по санкционированию («зеленом свете»). Она действительно хорошо выглядела, и уже там были определены некоторые творческие элементы. Инструменты, заложенные на тот момент, абсолютно не имели отношения к тому, что мы имеем в игре сейчас, но этого было достаточно, чтобы определить нужный вектор. На самом деле, пока мы не показали игру публике на КРИ, мы не знали, насколько людям интересна идея возможности создания собственных уровней и игр. Мне кажется, именно полученная тогда реакция укрепила нас в том, что мы делали.

Я был страшно рад, потому что сам всегда ратовал за идею разработки чего-то, что позволяло бы легко создавать собственные игры. Я увлекался этим со школьных времен, и все это время был в поиске идеального инструмента, в котором создавать игры было бы по-настоящему просто.

Надо иметь в виду, что все, кто пришел к нам, в особенности перед презентацией на КРИ, делали это на свой страх и риск, уходя с высокооплачиваемых должностей в наш клоунский балаган! И невзирая на трудные времена (у нас бывали тяжелейшие конфликты), поддерживать первоначальное воодушевление было совсем не сложно. Реакция, полученная на КРИ, значительно укрепила нашу уверенность, кроме того, нас мощно поддержало сообщество разработчиков игр и пресса, что было немаловажно. Карим оказался открытым и общительным человеком и отлично поддерживал общий боевой дух, что очень ценно, потому что я временами совершенный интраверт.

Эндрю: **Вы сказали, что как только команда вырастает в размерах, вы «скатываетесь в хаос». Объясните, пожалуйста, что это значит, и как вы с этим справились?**

Марк: Вы показываете что-то на экране, запускаете версии, с которыми можно «поиграть», и все начинают набрасывать свои идеи по поводу того, что с этим можно сделать. Если вы не сядете и не согласуете между собой общее направление, люди разойдутся каждый в своем направлении и начнут работать над идеями, противоречащими друг другу, и внезапно обнаружится, что больше нет единого целого и все разваливается на части.

Мне кажется, нам удалось достичь здесь хорошего баланса – официально я и Дэйв были ведущими разработчиками проекта, но на деле игру разрабатывали все вместе. И, разумеется, в итоге моя роль свелась к сбору идей других людей, осмыслению и переработке во что-то, так сказать, связное, а иногда к необходимости говорить: «Нет, этого мы делать не можем, потому что это бессмысленно по отношению к нашей общей цели». Понимаете, по сути, это попытка удержать общую концепцию, «видение», при использовании принципа постоянно эволюционирующего проектирования.

Эндрю: **Очень даже понимаем, и это перекликается с тем, что нам говорили многие: что необходимо зафиксировать концепцию в письменном виде, чтобы потом ее придерживаться. А что вы понимаете под «эволюционирующим проектированием»? Была ли эта эволюция постоянной? Были ли в вашей работе элементы итерации?**

Марк: Да, итерация – очень верное слово, по-моему. Конечно, у некоторых могут быть иллюзии насчет того, что можно спроектировать игру на бумаге, в виде рецепта, и потом отправить его на завод для изготовления. Для каких-то игр это, теоретически, возможно. Но по моему опыту, это никогда не срабатывает, уж точно не с играми, над которыми работал я. Скорее, у вас есть общая концепция – вроде неясного проволочного каркаса будущей статуи, куда каждый налепляет свой комок

глины, и когда вы пытаетесь как-то все это пригладить, она начинает приобретать определенный вид. Никто ничего не разрабатывает заранее. Понимаете, вы работаете, и вот, что-то начинает проявляться, а потом вы придаете этому некую форму.

Дженни: **Бывало ли так, что кто-то хотел вести работу над игрой в другом направлении, чем вы?**

Марк: Конечно, было много вещей, которые мы пробовали и потом решали: «Это не работает», – и мы возвращались и пробовали что-то еще. Поскольку мы довольно небольшая команда, то можем быть достаточно гибкими, чтобы внезапно поменять направление. Идеальная аналогия – разница в управлении скоростным катером и большим танкером. На катере вы можете изменить направление движения на раз. Танкер будет разворачиваться несоизмеримо дольше.

В начале проекта у нас были, скажем так, внутренние разногласия, будем ли мы делать платформер или креативный инструмент. Одни считали, что важнее разработать платформер, другие, наоборот, что инструмент для творчества; мне кажется, мы остановились посередине, что очень хорошо. Мы разработали потрясающий инструмент для творчества, а заодно и отличный платформер, который мог служить примером того, что можно сделать при помощи такого инструмента!

Должен признаться, иногда я добивался своего, будучи в очень плохом настроении или проявляя нетерпимость на работе. Звучит ужасно, но такое случалось. Бывали сложные моменты, иногда казалось, что все директоры просто до смерти ненавидят друг друга. Звучит неприятно, но, по-моему, хорошо, что так было. Это показывает, что нам всем небезразлично то, что мы делаем. Конечно, здесь не обойтись без эмоций и страстей. Но мне кажется, мы часто в результате приходили к самому верному решению (рис. 3.3).

Дженни: **Интересно, что речь зашла о страсти к продукту и конечной цели, потому что вы, похоже, были сосредоточены на том, чтобы создать очень правильную игру. Каким образом вы отслеживаете качество того, что создаете, особенно в условиях эволюции концепции?**

Марк: Просто мы все время в нее играем. То есть, по-моему, самое важное при создании игры – это как можно чаще в нее играть. Многие об этом забывают. Садишься, играешь и анализируешь: «Мне нравится? Меня увлекло? Это вот хорошо или раздражает? Скучновато? А что если бы здесь можно было вот так!» Мне кажется, целевой аудиторией должен изначально быть ты сам. Нужно делать что-то, что отвечает твоим внутренним стандартам для игры, в которую тебе хочется играть. А потом, если повезет, она понравится и другим. Однако если ты на-

Рис. 3.3. *Sackboy и Sackgirl – персонажи LittleBigPlanet.*

строился сделать что-то, чтобы поразить других, пожалуй, ты обречен на провал. Вот мой рецепт.

Еще очень важно тестировать игру на людях, которые ничего о ней не знают, там могут открыться очень неприятные истины, потому что сам ты можешь слишком углубиться в какие-то другие детали. Тебя может полностью захватить какая-то фишка в игре, а когда показываешь ее кому-то, кто никогда прежде в эту игру не играл, он ее даже не заметит. Зато внезапно откроет другую, тоже очевидную вещь. По-моему, это хорошее сочетание: постоянно играть самому и привлекать других. Тестирование, тестирование и еще раз тестирование, вот главный вывод!

Еще один важный фактор в плане качества – это правильный подбор людей. Поскольку мы небольшая команда и с самого начала знали, что хотим оставаться небольшой командой, мы очень осторожно относились к приему людей и наняли несколько невероятно талантливых специалистов. Это позволяет вам полностью доверять своим людям. По моему опыту, как только вас становится больше сотни человек, например, у вас начинается застой, появляются люди, которые делают что-то не потому, что они этим гордятся, а потому что получают зарплату. У нас такого нет. Здесь каждый действительно увлечен и гордится своей работой.

Эндрю: **И как вы находите нужных людей? Ведь это должны быть не просто умницы или те, у кого есть опыт разработки игр. Они должны идеально вписываться в команду, особенно если это маленькая команда.**

Марк: Это непросто. То есть в первый год существования, когда выпуск игры еще не был объявлен и прочее, а мы были начинающей компанией, все было еще в десять раз сложней, потому что хотелось найти опытных людей, тех, кто хорошо умеет это делать. И мы ожидали, что они уйдут с высокооплачиваемой работы и начнут все заново с людьми, которых даже не знают, что очень рискованно.

Первые шесть месяцев было тяжело; в основном, мы находили хороших людей, которые уже знали нас, и им было интересно то, что мы делаем. Но как только игра была заявлена на КРИ и получила очень хорошие отзывы, поиск персонала значительно облегчился.

Вдруг все очень заинтересовались нами и атаковали нас со всех сторон, но у нас была очень строгая процедура приема. В зависимости от конкретной позиции мы давали людям разные практические задания.

Что интересно, например, большинство набранных нами для разработки дизайна уровней сотрудников вообще не имели опыта работы в этой области. Тесты, разработанные нами для этих должностей, были в большей степени направлены на проверку их умения думать и творческих способностей. Для программистов, понятное дело, задания были более техническими, например мы давали им неисправную игру типа «космических захватчиков» и просили устранить баги и улучшить ее, сделать более классной.

Также мы давали кандидату пообщаться с другими людьми в команде. Можно довольно быстро почувствовать, будешь ты ладить с кем-то или нет. И, к счастью, мы делали хороший выбор. По-моему, только четверо из новеньких не справились и ушли после испытательного срока. То есть у нас неплохо получается.

Как только мы находили кого-то, кто был нам нужен, вопрос был только в том, чтобы наше предложение ему понравилось. У нас очень хорошая семейная атмосфера, здесь значим каждый человек. У нас нет винтиков. Каждый действительно важен в команде. Кроме того, у нас отличная система премиальных! Каждый, кто участвует в создании этой игры, если игра будет хорошо продаваться, тоже заработает много. По-моему, это самый честный подход, но в отрасли разработки компьютерных игр так поставлено далеко не везде. Очень часто люди работают на износ, а потом не получают достойного вознаграждения. Грустно, но это правда.

Эндрю: **Похоже, для вас очень много значит уважение к членам вашей команды.**

Марк: Оно необходимо. В прошлом я работал с разными командами, среди них были и плохие, и хорошие. Когда команда плохая, это просто

ужас. Люди вкладывают свою жизнь и умения, необходимо воздавать им по заслугам.

Эндрю: **У меня вопрос. Я немного ухожу в сторону, но хочу сказать, что мне очень нравится музыка. Я сам музыкант и меня просто поразила ваша подборка – она идеальная для этой игры.**

Марк: О да! У нас там есть великие мелодии.

Эндрю: **Я, правда, никогда прежде не слышал в играх такой музыки. Как появился этот саундтрек? Есть ли у него история? Как все это было?**

Марк: На нашем первом показе Sony и затем на ежемесячных совещаниях «Покажите-ка, что вы сделали» мы всегда пускали эту музыку с любыми показываемыми нами сюжетами. Это помогало создавать хорошую атмосферу. То есть оттуда все и развивалось логически. Когда мы начинали делать игру, мы задавали тон этими видеокадрами. Теперь, когда игру финансировала крупная фирма, нам говорили: «Мы можем получить лицензии на эти треки. Можете использовать их в игре».

То есть буквально вся команда начала выдвигать свои идеи по поводу песен и мелодий, которые им нравились, а мы потом передавали это в Sony, и они выясняли, могут ли получить на это лицензию или нет. Так все и было, честное слово.

Оригинальная интерактивная музыка была написана в сотрудничестве. Мы начали работать с еще одним другом, Мэтом, у него была студия и выход на многих настоящих музыкантов. Он кое-что для нас написал, а потом, сразу после КРИ, к нам пришел Кенни Янг и записал вместе с Мэтом все оригинальные треки. Кенни, кроме того, обработал лицензированные треки и создал все звуковые эффекты – отличный парень!

Дженни: **И вы заполучили Стивена Фрая на озвучку обучающего уровня?**

Марк: Да!

Дженни: **Вот это, действительно, круто. Как вам это удалось?**

Марк: Я очень хотел этого; с этой идеей креативного инструмента и интерфейса нам действительно нужно было разработать инструкции, я представлял это себе как видеоуроки и постоянно держал в голове: «Если кто-то будет учить меня, мне бы хотелось, чтобы это звучало интересно и смешно». Мне мгновенно пришел на ум фильм «Автостопом по галактике». Не знаю, смотрели ли вы этот фильм или читали книгу, но в фильме Стивен Фрай озвучивал саму книгу «Путеводитель автостопщика по галактике». И я думал: «Как прекрасно было бы заполучить его к нам», – даже не рассматривая это как возможность. Но кто-

то из Sony услышал об этом, и следующее, что я слышу: «Хорошо. У нас есть Стивен Фрай. Он согласен».

Следующее, что я помню, – я в студии, смотрю как эта огромная фигура читает скрипты к нашей игре! Мне довелось встретиться с героем! Несомненно, это одно из больших преимуществ работы с крупными фирмами. Они могут воплощать такие вещи. И нам очень повезло, что он действительно заинтересовался и полностью включился в игру. Так сбылась мечта, и мне кажется, его голос идеально подходит для этой озвучки.

Дженни: **Вы несколько раз сказали: я хотел этого и я получил это. То есть очевидно, что вы играли ведущую роль. В группе были разные роли, были люди, тянущие в разных направлениях – креативный инструмент или платформер, и вы направляли их. Считаете ли вы себя лидером? Что, по-вашему, характеризует хорошего лидера?**

Марк: Мне кажется, у проекта несколько лидеров. Я лишь один из них. И в любое время любой из нас может крикнуть громче других, если у него было больше эмоций по этому вопросу. И, конечно, были вещи, по поводу которых я впадал в отвратительное настроение и превращал жизнь всех на проекте в ад, покуда они не начинали делать по-моему. Но Алекс, Дэйв или Карим (или кто-то еще) могли страстно желать задать свое направление по другим вопросам. Мы собирались и спорили об этом, и очень часто чья-то позиция была гораздо лучше, чем другие, и они побеждали в этом споре...

То есть я бы не сказал, что играю здесь ведущую роль, но, пожалуй, я на проекте человек с самыми ужасными приступами плохого настроения. Это не очень приятная тактика, но она работает. Почти всегда.

Что касается хорошего лидера, это очень зависит от того, что вы возглавляете. Есть люди вроде Карима, который руководит художниками, – он отлично умеет ладить с людьми, никогда не применяет силовые приемы и притом чертовски хорошо рисует. Поэтому он может вести людей своим примером. Но мне кажется, лучшее умение руководителя – это в первую очередь умение набрать правильную команду и дать им возможность вести себя самим.

Дженни: **Если есть несколько лидеров, если от самой команды зависит больше – что делает команду хорошей? Если бы вы собирали, скажем, идеальную команду для создания игры или разработки программы, что бы вы искали? Не только в людях для команды, но и в самой команде?**

Марк: Нужны люди, которые замечательно разбираются в отдельных областях. Дэйв, например, невероятно хорош в физике и математике,

а Алекс очень силен в графическом программировании (а также в математике и графическом дизайне!). Мои ключевые навыки относятся к области технической графики. Карим хуже чем я в технике, но гораздо лучше меня в рисовании; каждую свободную минуту он рисует обнаженные тела. Есть целая группа совершенно необходимых людей, о которых я не сказал: нужен хороший специалист по кадрам, нужен кто-то, кто будет следить за рабочим окружением (поливать цветы, кормить людей, работать допоздна и т. д.). То есть нужны люди, которые очень хороши в своей области, но при этом интересуются и другими областями, способные и там внести свой вклад. Люди должны чувствовать большую ответственность за отведенную им часть игры/компании, но обладать достаточно широкими интересами, чтобы участвовать во всем. Наконец, что, пожалуй, важнее всего – взаимопонимание. Важно собрать вместе правильных людей; если в команде работают два гения, которые действуют друг другу на нервы, это катастрофа!

Эндрю: **Многие люди, включая меня, мечтают однажды сделать то, что сделали вы: создать замечательную игру, которую полюбят многие. У вас есть для них совет?**

Марк: Могу только сказать, что это было сложно, но принесло огромное удовлетворение. И, несомненно, я узнал о себе много нового. Я бы не стал недооценивать огромный объем тяжелой работы, необходимый для реализации подобного проекта, но также хочу сказать, что это точно возможно, и если вы не попробуете, это никогда не случится. Если вы действительно хотите это сделать, – начните и делайте, смотрите, что будет получаться, и преодолевайте трудности, не сдаваясь, когда дела пойдут неважно.

4

Построение идеальной команды

Билл Дипьер

Я хорошо помню, как подумал: «Э, да я не в курсе». Я сидел в конференц-зале Computech вместе с только что собранной командой высококвалифицированных разработчиков. Перед нами стояла задача привязать наши передовые программы по финансам из Microsoft QuickBasic к многообещающему новому продукту Microsoft – Visual Basic 3. Я как дока в Visual Basic должен был заниматься технической экспертизой и применить свой дорогой ценой полученный опыт к проекту, который определял будущее нашей фирмы.

Не верилось, что всего три недели назад я покинул крупный коммерческий банк, где отточил мастерство программирования. А с Джеком, моим нынешним боссом, познакомился всего пять недель назад. После того разговора я чувствовал доверие к Джеку, но очень сомневался, что мне удалось его поразить в профессиональном отношении. Ну и, конечно, труднее всего было поверить в то, что всего шесть недель тому назад я выложил 11 долларов 95 центов в книжном магазине Barnes & Noble за новенький экземпляр книги Билла Семпфа «Visual Basic for Dummies» (Wiley).

В конференц-зале в это время Джефф и Марк, два главных разработчика из нашей команды, обсуждали фундаментальные вопросы архитектуры проекта. Марк излагал свои аргументы: «Да знаю, что тут много дополнительной работы, но не вижу другого способа научиться делать пользовательский интерфейс, кроме как написать собственные элементы управления».

Джефф отвечал: «Марк, ты не прав. Ты знаешь, сколько сил на это надо? Я пишу на C++ намного дольше всех вас и вижу, что это нам не по плечу».

Но Марк не унимался: «Я просто не понимаю, какой еще может быть выбор. Панель управления Microsoft совершенно не подходит к нашему дизайну».

«Значит, надо менять дизайн».

«Нет, мы это не сделаем. Мы создаем технологии, чтобы дать клиентам то, что им нужно. Иначе можно было бы просто скопировать последнюю разработку Microsoft и предложить ее клиентам».

«Так почему бы нам не предложить им устройство для телепортации? Раз уж мы не ограничены в технологиях, так поможем им и с транспортом».

Видя, что диалог перестал быть конструктивным, Джек вмешался: «Итак. Прежде чем строить телепортер, вернемся к нашим баранам. Давайте разберемся. Ребята, мне прямо сейчас нужно ваше экспертное мнение. Джефф, прикинь – сколько времени займет то, что предлагает Марк?»

Джефф закатил глаза и громко вздохнул: «Да я сразу могу сказать, что мы сорвем все сроки».

Джек был невозмутим: «За это не беспокойся. Сроки – моя забота. Раз ты наш специалист по C++, прикинь эту идею и скажи, что думаешь».

Джефф кивнул: «Хорошо».

Обернувшись к Марку, Джек продолжал: «Марк, сделай мне, пожалуйста, список всех функций, которые можно сразу брать, без наладки». Марк не ответил, и Джек усилил нажим: «Помимо дат, мне надо позаботиться и о дизайне. Тебе надо будет работать с ребятами из Computech, так что скажи мне, что можно взять у Microsoft. Не волнуйся насчет дизайна, просто подумай, что можно сделать с готовыми функциями». Потом, почти незаметно подмигнув, он добавил: «И помни, я видел твой код и знаю, что ты можешь чертовски много».

Марк, не сдержав полуулыбки от комплимента, ответил: «Хорошо, Джек. Я посмотрю, что можно сделать».

Большинство из нас помнят 1990-е годы как расцвет разработчиков, так оно и было. Технический бум был в расцвете, и представители венчурного капитала сновали повсюду, выискивая, куда бы вложить свои деньги. И вкладывали они в программистов. Все мы помним истории о скалодромах и кофейных столиках за 10 000 долларов. И все это благодаря программистам. На них был большой спрос, и они могли дикто-

вать свои условия – даже если требовали в качестве гонорара скалодром или жутко дорогой кофейный столик.

В конференц-зале Computech мы собрались в тот день, когда взлет и падение технического бума еще только предстояли. Дело было в начале 90-х, и уже были подозрения, что создание программ – очень хороший способ заработать, но никто из нас и не представлял, насколько он хорош. Спрос на разработчиков вскоре быстро превысит предложение, а товарооборот индустрии достигнет рекордного уровня. Десять лет спустя в конференц-зале Computech я увижу те же лица, что и перед началом всей этой лихорадки. Это свидетельство командной солидарности. Никому из нас не заморочили голову скалодромы и кофейные столики. Собственно, не совсем так: Дуг ушел, но он не считается. Он уехал в турне с рок-группой, которой руководил. Я к этому еще вернусь.

В деловом плане Computech была достаточно раскрученной маркой, чтобы предложить конкурентоспособную продукцию тем покупателям, которые были заинтересованы в ней и готовы платить. Это была отличная стартовая позиция. Но с точки зрения технологии у Computech были серьезные ограничения. Она была написана на QuickBasic и создана для запуска на одном компьютере в DOS. Когда в финансовом бизнес-секторе стали развиваться компьютерные сети, все больше клиентов начали переходить на Windows. Им требовалась версия Computech для Windows. И мы хотели им ее предоставить. Расширенная поддержка сети, лучшее управление памятью и лучшие инструменты для разработки – далеко не все причины, по которым Computech тоже был необходим подобный переход. Не говоря о множестве совершенно новых возможностей интерфейса. Да что там – можно было пользоваться мышью!

И мы приступили. У нас было надежное, но ограниченное приложение для DOS. Покупатели требовали версии для Windows. И у нас было семь человек, которые должны были превратить одно в другое. Замечательный проект! Он был крупным, сложным, жизненно важным и ограниченным по времени. Нужно было сохранить с трудом завоеванную клиентскую базу, что означало почти полное отсутствие права на ошибку. Преимущества в то время казались огромными: этот проект вознес бы Computech к вершинам будущего. Это был классический программистский командный проект.

Конечно, если два руководителя проекта не ладят между собой, ни к чему хорошему это не приводит. После собрания моей главной задачей – неофициально – было понять, как разрешится эта борьба за власть. За беспристрастным компетентным мнением я обратился к человеку, который не был вовлечен в трения напрямую.

«Привет, Дуг, как оно?»

«Здорово, Билл. Ну, как тебе первый месяц?»

«Пока все хорошо. На собраниях так интересно!»

Дуг, казалось, немного удивился: «Ну да, наверное».

Гм, так дело не пойдет. Я решил приоткрыть карты: «Как думаешь, нам придется все писать на C++ с нуля?»

«Что? А, да не знаю. Джек разберется. Может, начнем, а там как пойдет».

Кажется, разговоры о телепортации не произвели на Дуга особого впечатления. Я не отступался: «Ну, а почему ты так думаешь?»

Дуг пожал плечами: «Не знаю, мне просто кажется, что он так и сделает. Он очень спокойный человек, так что обычно он просто что-то пробует, а дальше смотрит по результатам».

Это было неплохо, но я не очень-то понимал, в каком направлении развивается офисная политика. Сейчас я понимаю, что Дуг просто вряд ли мог пролить свет на динамику. Дуг был программистом не от природы, а по необходимости. Программированием он зарабатывал деньги для своего истинного призвания – карьеры рок-н-рольного антрепренера.

Дуг писал код со школы. Он пользовался пакетом MIDI, чтобы записывать музыку в гараже родителей на своем Macintosh Plus (где, разумеется, было целых 4 мегабайта RAM). К 27 годам он сменил Мак на синтезатор Yamaha. Его записи стали известны за пределами семьи. В то же время он был менеджером одной группы. Большую часть времени Дуг занимался музыкой, совершенно не скрывая этот факт на собеседованиях при приеме на работу. В чем работодатели видели помеху, то Дуг расценивал как достоинство. Это был сущий ребенок, сочетавший серьезные технические навыки с пренебрежением к иерархии. Дуг не собирался карабкаться вверх по карьерной лестнице или оставить след в мире программирования. Он просто хотел оплачивать счета, пока его группа не прославится.

Дуг написал много кода, он был полезен нашей команде. Но ввести меня в курс дела он не мог. Через пару дней я работал с Джеффом. Я обратился к нему за помощью со старыми кодами C++, которые нужно было экспортировать. Он очень меня выручил: в коде был ряд непонятных мне оптимизаторов, предназначение которых было очевидно для Джеффа. Я был впечатлен и сказал ему: «Джефф, это круто. Я бы неделю бился над тем, что ты объяснил мне за час».

Он почти любезно ответил: «Так я давно пишу на C и C++. Что-то уже делаешь чисто рефлекторно».

«Да, наверное. В любом случае – спасибо за помощь».

«Да не за что. Обращайся, если что».

Это меня порадовало – искренность, а не сарказм. Я решил продолжить: «Спасибо, обращусь. Так что с нашим пользовательским интерфейсом? Ты поговорил с Джеком?»

«Еще не успел, но поговорю. Это большая ошибка».

«Правда? А мне казалось, что у Джека есть свои резоны, учитывая возможности».

Джефф равнодушно пожал плечами: «Проблема Джека в том, что он считает, будто Марк может сделать что угодно, а это не так».

«Что ты имеешь в виду?»

«Я имею в виду, что Марк – главный специалист по пользовательским элементам управления, и это очень хорошо. Но у него нет и половины моего опыта по C, так что когда дело доходит до проекта вроде нашего, то он ни в чем не разберется. Не думаю, что Джек это понимает».

«Да, возможно. Наверное, Джек просто думает, что стоит рискнуть».

Равнодушие Джека было нарушено. Он заерзал в кресле и сказал: «Послушай, ты ведь специалист в Visual Basic – и говоришь, что без меня не можешь разобраться. Это чертовски сложно. То есть я могу это сделать, но я уже столько времени над этим работаю. Просто безумие браться еще и за это помимо всего, что уже надо сделать, когда есть решение, которое выполнит за нас 99% работы».

Вообще-то я не говорил, что без него не обойдусь. Я только сказал, что у меня бы это отняло больше времени. Ну да ладно. Но мне действительно нужно было понять причины недовольства Джеффа Марком, и я решил оставить в покое личности и перейти к статистике. Стараясь поддержать объективный тон, я сказал: «В общем да, но это не 99%, скорее 59%».

«Да хоть так, все равно по сравнению с тем, что клиенты имеют сейчас, это большой шаг вперед».

Тогда я пустил в ход аргумент из серии «верный солдат»: «Что ж, если Джек сказал, что это его забота, то не будем мешать ему заботиться».

Но Джефф был не согласен: «Джек может заботиться сколько угодно, но если он думает, что Марк сядет и начнет строчить код на C++, то он ошибается».

Кажется, кто-то вышел из себя. Я попытался дать логичный ответ: «Не знаю, Джефф. Насколько я могу судить, Марк чертовски хороший разработчик».

«Естественно, Марк может писать код. Он хорош в том, чем занимается, но не надо требовать от него слишком многого».

Да, Джефф был очень чувствителен, когда речь шла о Марке. Я не сдержался и спросил:

«Почему? Ведь код есть код. А Марк умный парень».

«Ну, *умный* – понятие относительное».

Тут Джефф полез в задний карман за бумажником. Порывшись в нем, он вручил мне бережно заламинированную карточку. Оказалось, что Джефф состоял в Менсе – собственно, это был членский билет.

Да уж, вот это перемена – от искреннего и готового помочь человека до полного мудака всего за какие-то пять минут. Я был потрясен, но сумел пробормотать: «Да, впечатляет, но все же подозреваю, что ты недооцениваешь Марка».

Джефф пожал плечами и не очень уверенно произнес: «Гммм, возможно. Посмотрим».

Итак, этот парень может писать код на C++ вслепую, но как член команды имеет серьезные недостатки. Джефф определенно считал, что Джек слишком полагается на Марка, и это его раздражало. Я работал во многих проектах и мог понять, когда кто-то начинает сходить с рельсов. Этот парень на полной скорости катил прямиком к обрыву.

На следующее утро я пил кофе невдалеке от кабинета Марка и решил взглянуть, над чем он работает. Довольно часто это помогало мне ускорить программирование. В тот самый день мы обсуждали модель событий, которую Марк разрабатывал для нашего приложения. В ранних версиях Visual Basic не было общих методов для форм, в результате работать с MDI-приложениями было неудобно. Марк же предложил простой и эффективный обходной маневр, использующий свойства ярлыка скрытой кнопки для имитации событий в формах. Это было типичное решение Марка – элегантное и простое в применении, какое нам и требовалось.

Мы с Марком быстро установили хорошие взаимоотношения. Он как разработчик был компетентнее меня, но на программирование мы смотрели сходно. Я поддерживал его недавнее утверждение о том, что прежде всего надо сосредоточиться на нуждах клиентов, а не на технологии. Меня интересовало, что он думает о конфликте с Джеффом.

«Ну что, Марк, как идет работа над проектом с Microsoft? Как думаешь, получится у нас обойтись без собственных доделок?»

«Только если пожертвовать требованиями дизайна. Джефф как-то слишком против этого. А по-моему, было бы круто сделать собственные элементы управления».

Честно говоря, я никогда не задумывался о том, что собственное удовольствие тоже может играть важную роль в принятии решений. Конечно, я как разработчик вряд ли мог не согласиться: это был бы отличный проект. Но я постарался войти в положение Джека и сказал: «Да, было бы забавно. Но не уверен, что для Джека сейчас это главное».

«Да, но пусть держит это в голове. Если работники трудятся с радостью, то это уже полдела». Его улыбка не была ни заговорщицкой, ни издевательской. Марк продолжал: «В любом случае Джефф ведет себя так, будто только он один во всем в мире может писать на C. Погодите. Если мы напишем все сами, то это точно себя оправдает. Не так уж много времени это займет».

Осторожно, впереди обрыв. «Ну, если это не отнимет больше времени, то все просто. Думаю, поэтому Джек и хочет это время прикинуть».

Тут как раз появился Джек. «Что я хочу прикинуть?» Как руководитель Джек был исключительно доступен. Нередко он сам подходил и разговаривал с людьми. Беседы эти всегда были неформальными и часто не имели прямого отношения к работе. Но сейчас он вклинился прямо в рабочее обсуждение.

Я ответил ему: «Мы с Марком выясняли, за какое время сможем сами сделать элементы интерфейса».

Джек кивнул и сказал, глядя на Марка: «Кстати, я как раз хотел с тобой это обсудить. Сейчас мне некогда, так что давай завтра утром, в 10».

«Конечно».

Потом Джек повернулся ко мне и добавил: «Билл, почему бы тебе тоже не прийти? Хочу, чтобы и ты полностью был в курсе».

«Хорошо, Джек».

«Отлично, значит завтра в 10». И вышел в коридор. Я посмотрел на Марка, но он был непроницаем. Тогда я сказал: «Что ж, скоро мы узнаем ответ».

«Да. Надеюсь, правильный».

Я тоже надеялся, но мне стало не до того. Надо было решать свои задачи. В Computech я отчасти был менеджером – для одного разработчика. Боб работал в Computech семь или восемь лет. На первом собрании я удивился, почему он должен подчиняться мне, а не наоборот. Но вскоре это прояснилось.

Один из наших техников пришел ко мне в кабинет и сказал, что у Боба кончился тонер в новом цветном принтере на следующий день после замены картриджей. Техник расстроился не на шутку: новый дорогой тонер нужно было заказывать заранее. Более того, техник намекал, что

Боб израсходовал его без надобности. Мне как менеджеру Боба предстояло раскрыть тайну Тонергейта.

Я подошел к отсеку Боба. Начав с разработки, я перешел к истинной цели своего визита: «Боб, ты много печатал? Похоже, в нашем LaserJet за сутки кончился тонер».

У Боба были недостатки, но нечестность в их число не входила. Так что он ответил: «Да».

Разговорчивостью Боб тоже не отличался. Я продолжал: «А зачем? Зачем ты столько печатал?»

«Я печатал меню для нового тайского ресторана моей девушки. Помнишь, я дал тебе визитку?»

Я, конечно, помнил, что Боб дал мне визитку, рассказав о своей новой девушке и ее новом предприятии. Вот только я не помнил, чтобы Computech обещала предоставить свои ресурсы для этого проекта. «Да-да, припоминаю. А почему ты печатаешь меню здесь?»

«Потому что в типографии слишком дорого».

Что ж, в логике Бобу не отказать. Но в других областях он был слабее – например в общественных условностях или в понимании вещей, очевидных для других. «Да, Боб, но нельзя печатать в офисе просто потому, что ты не хочешь за это платить».

«А почему? Мне казалось, что нам можно использовать принтер и для личных нужд».

«С чего ты взял?»

«Ну, на прошлой неделе Марк тут печатал фотографию своей дочери».

«Да, но всего одну. А ты истратил все чернила. Теперь всю неделю никто ничего не сможет печатать, пока нам не привезут тонер».

«Я не знал, что краска кончится».

«Но она кончилась. Не надо было столько печатать».

«Ну, если есть какие-то ограничения для личных нужд, то надо было мне об этом сказать».

Тут я взорвался: «С этого момента тебе вообще запрещается что-то печатать!»

Теперь взорвался уже Боб: «Но это несправедливо! Почему для меня другие условия, чем для остальных?»

«Потому что остальные другие, Боб. Условия не всегда одинаковы для всех».

«Отстой какой. Я поговорю с Джеком».

Я вздохнул: «Прекрасно, я тоже с ним поговорю».

«Отлично», – ответил Боб и отвернулся к монитору.

Это был непорядок. Боб совсем не был великим разработчиком. И с ним явно было тяжело работать. Но его труд был нам нужен, и я как новый человек совершенно не хотел сразу портить с ним отношения. Моя задача менеджера состояла в том, чтобы сплотить команду, а не поссорить всех между собой. Я ходил по кабинету, вновь и вновь вспоминая эту сцену и думая, как можно было действовать по-другому. Иного пути я не видел, но все же с тревогой ожидал, что скажет Джек, когда узнает, как я разговаривал с Бобом.

К счастью, мне было не до беспокойства, потому что надо было успеть на встречу в среду в 11 утра. На этом еженедельном собрании должны были присутствовать практически все члены команды. Обычно оно проходило без особых событий, но учитывая напряжение между Марком и Джеффом, а теперь еще и мной и Бобом, я слегка волновался. Я собирался подсесть к Дугу, поскольку решил, что в случае стычки он, скорее всего, устранится. Но сев, я тут же пожалел о своем решении. С первого взгляда стало ясно, что Дуг тоже нервничает. Он облизывался и ерзал, барабанил пальцами по столу. Взгляд его блуждал по комнате. Я собрался с духом. Если большинство участников собрания, включая меня самого, уже на грани, то ничего хорошего явно не выйдет.

Дуг начал разговор, по-прежнему вертясь на кресле: «Так, я все же хочу кое-что выяснить, потому с трудом понимаю, что происходит».

Началось. Я обвел взглядом комнату в ожидании реакции. Все, казалось, были слегка удивлены. Непонятно, против кого же будет направлено обвинение.

Дуг продолжал: «У меня сегодня тяжелый день, потому что вчера вечером на концерте в церкви наш солист выругался матом».

Послышался сдавленный смешок. Я почесал рот, чтобы скрыть улыбку. Однако Дуг, похоже, не осознав, что мы не разделяем его расстройство, стал описывать карьерные последствия этого непостижимого ложного шага. Я посмотрел на Джека, который, как и все, пытался подавить ухмылку. В то время я не знал, что Джек сознательно шел на риск, приняв на работу парня, чьи долгосрочные планы не касались Computech. Сейчас я думаю, не относилась ли улыбка Джека к тому, что он понимал все выгоды от такого риска.

Пару часов спустя я работал над каким-то кодом у себя в кабинете, когда зазвонил телефон. Я снял трубку: «Привет, это Билл».

«Билл, это Джек. Есть минутка?»

«Да, конечно, а что такое?»

«Можешь ко мне зайти? У меня тут Боб, я бы хотел поговорить насчет этого принтера».

«Да, конечно, сейчас приду».

Повесив трубку, я чертыхнулся. Надо было обязательно дойти до Джека и изложить свою версию случившегося раньше. А я промедлил, и Боб меня опередил. Джек с Бобом работали вместе почти десять лет. Я же был знаком с Джеком меньше месяца.

Когда я вошел в кабинет Джека, он посмотрел на меня из-за стола и сказал: «Привет, Билл, спасибо, что пришел». И добавил, неопределенно махнув рукой: «Садись». Боб уже сидел перед столом Джека, я сел на другой стул. При этом я кивнул: «Привет, Боб».

Чтобы не уступать мне, он сдержанно ответил: «Привет, Билл».

Уже что-то вроде беседы. Не тратя время, Джек сразу перешел к делу: «Так вот, Билл, я тут поговорил немного с Бобом насчет принтера. Он считает, что ты несправедлив к нему».

Я все утро обдумывал, как лучше все представить. Наконец, я решил, что факты должны говорить сами за себя. В другой ситуации я бы попытался быть более дипломатичным. Но в Computech, к сожалению, я мало что знал, поэтому это особо не помогло бы. Поэтому я еще раз изложил, как все произошло, ничего не приукрашивая. И под конец добавил: «К тому же я узнал, что на эти меню Боб израсходовал четыре пачки бумаги. Я знаю, что бумаги у нас много, но, по-моему, это в принципе недопустимо».

Боб был к этому готов и ответил: «Да с тех пор как я здесь работаю, я все время экономлю бумагу. Только я всегда переворачиваю лист и печатаю на обороте. Не считал, но за последние шесть лет я сэкономил для Computech, наверное, пачек десять бумаги».

Точно не уверен, но, кажется, в тот момент у меня отвисла челюсть. Джек никак не мог ведь на это купиться, правда? Это было просто смешно. Но тут была внутренняя логика, к тому же Боб работал здесь уже давно. Можно было догадаться, что для Джека подобный подход не был сюрпризом. Неужели он смотрел на это сквозь пальцы? Когда Джек заговорил, я вцепился в подлокотники.

«Извини, Боб, но тут я на стороне Билла».

Я облегченно вздохнул и ослабил хватку. Боб начал было возражать, но Джек прервал его: «Больше ни у кого нет таких проблем. Все, кроме тебя, похоже, понимают, что есть какие-то разумные пределы. Ты просил назвать предел – вот тебе предел, живи теперь с ним».

Боб продолжал протестовать: «Но это неправильно, это несправедливо!»

Джек ответил фразой, которую я за эти годы слышал от него еще много раз: «Если бы жизнь была справедливой, то не было бы богатых». Довольно странное высказывание, учитывая, что большинство наших клиентов были воротилами с Уолл-стрит, но оно возымело свое действие и положило конец протестам.

На том, собственно, и порешили. Позже в тот же день я заглянул к Джеку поблагодарить его за поддержку. Он ответил с теплой улыбкой: «Да ладно. Я знаю, что с Бобом могут быть проблемы. Ты сделал все, что мог».

Я спросил: «И поэтому ты отправил проблемного парня к новичку, чтобы избавить от него всех остальных?»

Джек ухмыльнулся: «Или же я отправил новичка к нему, чтобы посмотреть, как новичок с ним управится».

Теперь настала моя очередь улыбаться: «Ну и как я, справился?»

«У тебя получилось».

В последующие годы я часто думал о Бобе и о том, зачем Джек держал его в фирме. Конечно, он был отличной лакмусовой бумажкой для новых менеджеров, но вряд ли это оправдывало его зарплату. Так что же спасало Боба? Какая черта делала его важным винтиком в хорошо смазанном механизме? Посредственное умение кодировать? Его трудности с общением? Думаю, ничто из этого. Боб вовсе не был важным винтиком. Главным было то, что Боб был частью нашей команды. Боб отнимал у Джека кучу времени, но Джек и бровью не вел. Джек всегда мог нас поддержать, облегчить нашу работу. Это он хотел донести и до Боба, и до всей нашей команды.

Я уже говорил о том чувстве солидарности, которое вызывала в нас Computech. Эта солидарность порождалась не электронными письмами управляющих и не корпоративными обедами. Солидарность – улица с двусторонним движением. Если менеджер согласен работать даже с самым трудным своим подчиненным, то какие чувства это вызывает у всех остальных сотрудников? Вспоминаю, как через несколько лет, когда Computech уже купили, а затем новая фирма-хозяин сама была куплена конкурентами, началась череда сокращений. Сокращения закончились в декабре, и в следующем феврале мы уже наслушались от нашего нового руководства о производственной чести и моральных качествах. Как это типично для мира слияний и поглощений!..

Я подумал о Бобе, которого освободили от должности из-за низкой эффективности. В каком-то смысле я был рад избавиться от него. Видит бог, с ним было нелегко. Нельзя не согласиться, что и в деловом плане были серьезные причины его уволить. Или нет? В этих ежедневных электронных письмах, которые нам рассылали для поддержки кол-

лективного духа и которые мы осмеивали, было много уловок, дабы оставить нас послушными и продуктивными. Но производительность только падала. Боевой дух тоже. И я подумал о том, что бы случилось, если бы такое письмо разослал нам Джек, попросив затянуть пояса во имя Computech. Думаю, все мы поверили бы ему и выполнили его просьбу. Мы пожертвовали бы чем-то ради него, потому что знали, что он сам сделает это для нас, если понадобится. И это знание принес нашей команде Боб: он показал, что мы действительно команда. Он был, наверное, самым очевидным инструментом Джека, который доказывал, что мы должны быть вместе во всем.

На следующее утро я зашел в кабинет Марка чуть позже обычного. Обсудив несколько технических вопросов новой архитектуры, мы пошли к Джеку. Джек поприветствовал нас в своем кабинете, мы расселись по местам и стали обсуждать футбольную команду «Филадельфия иглз». Джек ставил на Рэндалла Каннингема, а мы с Марком считали, что его лучшие годы уже позади. Эта тема (которую мы поднимали несколько раз в неделю) иссякла сама по себе, и Джек заговорил о другом.

«Итак, поговорим об интерфейсе. Марк, что ты думаешь? Много ли нам могут дать элементы управления Microsoft?»

Марк ответил: «Да, я несколько дней изучал вопрос и, похоже, они удовлетворяют от половины до двух третей наших потребностей». Он вкратце упомянул те функции, которые нельзя реализовать таким образом. Джек задал несколько специальных вопросов о том, какие требования дизайна необходимы, а какими можно и пожертвовать. Марк, поскольку был в теме, ответил на все вопросы. Вскоре выяснилось, что элементы управления Microsoft покрывают по меньшей мере две трети наших потребностей, даже 75–80%.

Если Джек и заметил неувязку, то вида не подал. Вместо этого он сказал Марку: «Ладно, давай напишем один элемент управления с нуля как пилотный вариант. Посмотрим, что у нас выйдет, и потом уже решим с остальными». Логика такого управленческого решения была прозрачна. Собственно говоря, у меня сложилось впечатление, что Джек принял это решение еще до того, как мы вошли тем утром в офис.

Не знаю, думал ли Марк так же, но он точно обрадовался такому решению: «Отлично. Который берем?»

Это было уже легче. Имело смысл сначала сделать что-то самое простое, избегая сложностей в пилоте. Я бы выбрал кнопку. Однако Джек наморщил лоб, как будто о таких подробностях еще не задумывался. И спросил: «Ну, не знаю, а ты сам какой бы выбрал?»

Марк просиял: «Думаю, было бы неплохо начать с виртуального списка. Это сложно, но зато мы сможем применять его везде, так что это

самое эффективное». Помедлив, он добавил: «К тому же его будет интересно писать».

При виде такого энтузиазма Джек не сдержал улыбку: «Хорошо, пусть будет виртуальный список. Сможешь к концу завтрашнего дня оценить время? Кроме того, мне нужно, чтобы ты работал с Джеффом». Марк скривился. Джек продолжал: «Не беспокойся, я с ним поговорю. Но надо воспользоваться его возможностями, если мы хотим уложиться в нормальные сроки. Иначе придется удовольствоваться Microsoft».

Мы втроем поболтали еще несколько минут. Выйдя из кабинета Джека, я повернулся к Марку и сказал: «Отлично, парень! Ты сделаешь этот список».

Марк опять просиял. Потом посерьезнел и сказал: «Да, и я собираюсь сделать его так, что он доверит мне и все остальное».

Конечно, Марк был только одной частью уравнения. Необходима была и помощь Джеффа. Для этого Джеку нужно было придумать что-то похитрее. Позже в тот же день Джек встретил Джеффа, чтобы поговорить с ним один на один. Щекотать чье-то эго Джек предпочитал тет-а-тет. Поэтому в тот раз меня не было в кабинете Джека, но я потом расспросил его, так что зная, как он обычно работает, могу легко восстановить их разговор. Сначала Джек польстил самолюбию Джеффа, показав ему важность его мнения, примерно так: «Спасибо, Джефф, что зашел. Я хотел еще раз использовать твои мозги в вопросе об элементах управления. Как ты считаешь, трудно будет сделать свои?»

Джефф, по-прежнему держа оборону, ответил: «Ну, как я уже сказал, будет куча дополнительной работы. Думаю, не меньше трех месяцев». И сделал паузу, чтобы подчеркнуть неопровержимость своих слов. Джек, как всегда невозмутимый, утвердительно кивнул, и Джефф продолжил: «Не понимаю, зачем все это. Мы не получим особой выгоды, потратив время на разработку».

Джек начал крушить защиту Джеффа исподволь: «Я понял. Давай пока забудем о времени на разработку. Меня больше интересует ее выполнимость». Потом он разбил главный, хотя и не высказанный аргумент Джеффа: «Просто я считаю, что это будет полезный для команды опыт, в особенности для Марка. Мы оба знаем, что программист он хороший, но лучший в C++ у нас – это ты. И думаю, наша команда станет сильнее, если мы распространим твои умения и среди остальных».

Отчасти обезоруженный, Джефф ответил: «Ну да, конечно, всегда полезно, когда знаниями обладает вся команда, но не уверен, есть ли у нас для этого время».

Здесь Джек уже мог сослаться на собственный опыт. «Ну, тут дело в том, что для этого подходящего времени просто не бывает. Мне не до-

водилось участвовать в проектах, где давалось бы время на подготовку команды. Но я убедился, что дополнительный опыт всегда оправдывается по окончании проекта».

Почувствовав, что его мастерство не подвергается сомнению, Джефф признал компетентность Джека в управлении проектами: «Ну ладно. Если ты считаешь, что отсрочка – это нормально, тогда поступай, как хочешь».

Теперь было легче убедить Джеффа изменить планы, и Джек решил купить его с потрохами: «Да, так и сделаю. Над этим тебе потеть не надо. Придется попотеть только с помощью Марку, если она ему понадобится. Как я говорил, надо распространять твои знания. Что на это скажешь?»

«Да мне-то все нормально, Джек. Просто я не уверен, что это будет эффективно».

«Ну так что? Я дам вам час в неделю, чтобы работать над этим вместе, ну, чтобы как-то делиться идеями. Попробуем пару недель и посмотрим, как получается. И попробуй быть терпимее с Марком; он привык, что знает больше остальных, так что ему может быть трудно разговаривать с тобой о C++».

Джек знал, что делал. Джефф просиял и утвердительно кивнул.

«Спасибо, Джек, я оценил».

«Конечно, без проблем».

Через несколько дней состоялась первая из таких встреч. Сначала Марк и Джефф каждую неделю кодировали список на C++. В разгар работы над проектом они устраивали импровизированные совещания трижды или четырежды в неделю, а закончив со списком, перешли к кнопкам, текстовым окнам и другим вещам, сделав с полдюжины пользовательских элементов управления к концу проекта. Предсказание Джеффа о трехмесячной задержке не сбылось: потребовалось всего два месяца, главным образом потому, что в процессе работы они получили необходимый опыт, а знания Джеффа, которые он передал Марку, распространились и среди других членов коллектива.

А я, многообещающий эксперт по Visual Basic, теперь вижу, что мои тревоги по поводу того, что я «не в курсе», были беспочвенными. Я вполне разобрался в динамике развития нашей команды. Никаких тайн, оказывается, не было. Мне на это уже намекали, но большую часть года я пытался в это поверить. Отлично звучит! Команда, сконцентрированная на работе и поддерживающая друг друга в едином стремлении к одной цели. Да, так и было. Не то чтобы это были 10 лучших программистов, когда-либо работавших вместе. Но каждый из нас привносил в коллектив что-то свое. Каждый из нас делал свой вклад в общий труд,

нами искусно управлял лидер, неравнодушный к людям, осведомленный об их сильных сторонах и готовый мириться с их недостатками.

Итак, вы уже могли догадаться, что переход Computech с QuickBasic на Visual Basic 3 имел большой успех. Да, так и вышло. Перед нами вставали препятствия. Например, Дуг таки добился успеха. И даже крупного. Его группа отправилась в турне по США. К счастью, проект уже подходил к концу, так что когда он ушел, его уже можно было заменить. Боб вносил лепту в общее напряжение, некоторые из его функций так и не попали в первый выпуск для Windows. Но это были лишь отдельные недостатки на общем фоне ошеломляющего успеха.

Через десять месяцев после того дня в конференц-зале первые пользователи Computech получили версию нашей программы для Windows. Реакция была почти единодушно положительной, сообщений об ошибках приходило мало, продажи зашкаливали, дирекция была счастлива. Команда разработчиков получила свою долю восторгов. До сих пор я порой вспоминаю эти дни с ветеранами Computech, и они говорят, что нам посчастливилось собрать так много замечательных разработчиков в одно и то же время на один проект. Я всегда отвечаю, что это была не удача и не замечательные разработчики; это была идеальная команда.

5

Что мотивирует разработчиков

Интервью с Энди Лестером

Мы решили узнать, что мотивирует разработчиков, и поиски привели нас к Энди Лестеру. Он настоящий эксперт по найму умных людей и построению работоспособных команд, много писал и выступал на эти темы. Как член команды разработчиков Perl он знает, чего стоит создать выдающиеся программы. У него мы решили узнать о причинах, которые мотивируют – и наоборот! – тех, кто создает программы, и о том, как помочь им работать вместе.

Эндрю: **Откуда вы знаете так много о командах разработчиков и их работе?**

Энди: Я профессионально программирую 21 или 22 года, работал в хороших и в плохих командах. Прежде всего я программирую ради удовольствия. Я и так могу заработать, и заработать неплохо, тем, что люблю делать. Если бы мне работа была не в радость, она бы того не стоила. И часть этой радости исходит от тех, с кем я работаю. Конечно, хорошо работать и одному, но аспект общения – и на работе, и в проектах с открытым кодом – нельзя отрицать. Мир открытого исходного кода непосредственно связан с общением, потому что иначе ты писал бы программы только для себя.

Эндрю: **Как вы считаете, этот аспект играет большую роль в повышении производительности команды? Потребность в общении – это то, что испытывают все члены команды, а не отдельные ее представители?**

Энди: Трудно сказать. Вспомните пирамиду потребностей Маслоу: социальные потребности находятся в самом низу пирамиды, то есть они одни из основных. Но общение для разных людей означает разные вещи.

Дженни: **То есть хорошей команде необходимо дважды в неделю вместе посещать пиццерию? Что для вас «общение»?**

Энди: Для кого-то «общение» – это нечто отдельное от той работы, которую они выполняют. Так что да, мы можем вместе есть пиццу или еще как-то обедать вместе. А для некоторых социальный аспект полностью связан с работой: «Хочу работать вместе с другими». Кто-то любит надевать наушники: «Не беспокойте меня ближайшие четыре часа». И тут нет правых и виноватых, это уж кто как привык.

Эндрю: **Таким образом, нужно быть чутким к тому, как работают люди в вашей команде, и давать им наиболее комфортную среду и самое интересное окружение?**

Энди: Да. И те, кто попадает в новую для себя среду, должны понимать динамику сложившейся команды. Если ко мне, Энди, кто-то подойдет и скажет: «Слушай, у меня тут проблема – я не понимаю, что вот тут не так с этой функцией», – то я буду рад помочь, но некоторые воспримут это как грубое посягательство на их рабочее время. Я считаю, что помощь другим – такая же часть моей жизни, как и собственно написание кода. Но так думают не все.

И это трудно, потому что мы настолько привыкли к собственному образу мышления, что первая реакция на возникающие конфликты – счесть другого неправым. «Зачем ты меня беспокоишь? Ты неправ». А я

вот считаю, что неправ тот, кто бьется над кодом в одиночку, не привлекая окружающих, я обязательно подойду к нему и скажу: «Эй, я бы мог тебе помочь». Когда вы понимаете внутреннюю жизнь команды, в которую попали, это придает уверенности в себе и наблюдательности.

Эндрю: **Как вы пришли к этому? Через эксперименты? У вас был печальный опыт?**

Энди: Мне случалось работать в таких командах, где я, коммуникабельный Энди, считающий, что общение – это сама жизнь, был решительно неуместен. Где вмешательство в работу считается главным грехом. Мне было неуютно. Из-за этого постоянно возникали конфликты, и я чувствовал себя одиноким и оторванным от коллектива, причем для других я, несомненно, был головной болью.

Дженни: **Да, но как быть с теми, кто раз узнав, что им могут помочь другие, стремится постоянно опираться на них? Кто, столкнувшись с первыми трудностями, обращается за их разрешением не к Гуглу, а к коллегам?**

Энди: Что ж, то, как вы это формулируете, уже говорит, что вы об этом думаете.

Дженни: **Ну, вот я и спрашиваю: может, это неправильно?**

Энди: Иногда да. Все зависит от общественных норм. Дело в том, что программисты специфически относятся к нормам общества. И я не о чем-то вроде «эй, мылся бы ты почаще». Я отношусь к этому юмористически, но и вместе с тем со всей серьезностью.

Я говорю об общественных нормах любой заданной группы. И они могут быть очень строгими: надо обращать внимание на окружающих. Недостаток такого внимания может сказаться отрицательно. Вот, например, вчерашняя история, когда мы проверяли код. Трое из нас уже набили руку в проверке кода. Четвертый, чей код как раз и проверяли, чувствовал себя в этой компании куда менее комфортно. Мы стали разговаривать о коде, обсуждать его и так далее. И чем больше мы углублялись, тем неприятнее становилось парню, код которого проверялся. Один из ребят все больше возбуждался, все более оживленно излагал свою точку зрения, и ему было наплевать, что чувствует автор кода. А я смотрел на него и замечал, что он все более подавлен таким ходом вещей. Двое остальных не обращали на него никакого внимания, они просто разговаривали друг с другом, и такое невнимание становилось отвратительным. Я понял, что происходит, и сказал: «Знаете, ребята, мы уже давно сидим. Давайте-ка прервемся». Чтобы ослабить давление.

Гики[1] – а к ним я отношу и себя – по своей природе не могут оценить масштаб аудитории. А их аудитория – это обычно все, кто может их услышать. В данном случае нас за столом было четверо, но постепенно разговор превратился в диалог двоих, которые не обращали внимания на других людей в комнате. Если бы я их не остановил, могла бы возникнуть настоящая враждебность.

Эндрю: **Знаете, я из таких же гиков.**

Энди: Да благословит вас Бог.

Эндрю: **И мне потребовалось много времени, чтобы это осознать и изменить манеру общения с людьми. Одной из моих привычек был – причем в молодости это проявлялось куда сильнее – чрезмерный педантизм, в том случае когда я осознавал свою правоту. Я был готов спорить по этому поводу до конца, не интересуясь, насколько оппоненту хочется продолжать спор. И только очень нескоро я понял, что не стоит делать слишком широких обобщений и держать спор в русле конкретной проблемы, потому что остальные, не гики, могут посчитать такую манеру крайне раздражающей.**

Сейчас у меня много друзей-гиков. У них те же проблемы, но они, кажется, даже не осознают их. Собственно, они бы очень удивились, если бы им сказали, что обобщения и споры по их поводу – проблема, а не предпочтительный способ общаться с другими. Что делать, если в вашей команде попадается такой человек? Как помочь этим людям? И нужно ли им вообще помогать? Может быть, вы пытаетесь изменить среду, чтобы она им больше подходила?

Энди: Многое зависит от ваших взаимоотношений с таким человеком. Многим гикам «приспосабливаемость» – вот уродливое слово! – не только не нужна, но и считается у них признаком слабости. Многие гики гордятся тем, что они не похожи на других. Отличие – повод задрать нос. Думаю, само по себе это хорошо, если ты действительно хочешь выделяться. И совсем другое дело, если это вредит коллективу.

Люди либо вливаются в команду, либо нет. Помощь тем, у кого не получается, может оказаться трудным делом, потому что столь важная для всех людей вещь, как потребность в понимании, уважении, возможность быть выслушанным, может восприниматься гиками как слабость. Тогда борьба будет очень трудной.

Дженни: **Получается, что в программировании задействованы только такие люди. Неужели это правда?**

---

[1] Гик (англ. geek) – эксцентричный человек, одержимый технологиями, в том числе компьютерными. – *Прим. ред.*

Энди: Нет, конечно. Не все мы социально не адаптированы.

Дженни: **Хорошо.**

Энди: Но довольно многие действительно беспомощны в обществе, они выросли, общаясь с компьютерами – и предпочитая общаться с компьютерами, потому что компьютер делает то, что ты хочешь, он общается с тобой точно, логически. Мне не надо думать о том, как компьютер себя чувствует. Я могу рассердиться, обозвать компьютер, и компьютер на меня за это не обидится. С ним не бывает таких отрицательных эмоций, как при общении с людьми. Спросите любого, кто живет в браке или с постоянным партнером. Да, такие эмоции могут возникать. Но этого не избежать, общаясь с другими людьми.

Дженни: **И как же наступает согласие? Меня интересуют два аспекта. Один – это то, как люди вообще реагируют на критику. Когда вы рассказали о проверке кода, мне показалось, что это история о том, что люди должны быть внимательнее друг к другу. Другой аспект этой истории в том, что в команде нужно стремиться всем помогать – с кодом, с работой – и позволить другим реагировать на это, принять эту реакцию и изменить что-то в связи с этим.**

Энди: Один из случаев – когда среди вас есть кто-то, кто может изолироваться – я употребляю мужской род, потому что обычно это мужчина, – логически отделить себя от своего кода, своей работы. Допустим, кто-то подходит к нему и говорит: «Эй, слушай, твой код отстой, потому что ты делаешь так, так и так». И получает логически отстраненный ответ: «Хорошо, я просто не думал об этом, теперь подумаю». А другой гик или даже обычный человек воспримет это как нападение, как удар лично по нему. «Ты сделал это неправильно», – обычная формулировка в подобном случае. И кто-то может воспринять это довольно остро.

Допустим, один компьютерщик проходит по отделу, видит кого-то в другой комнате и говорит: «У тебя таблица неправильная, надо было сделать так, так и так». И он действует строго с логической точки зрения: «Этот продукт может быть лучше, если его изменить». В то же время человек, работа которого сделана, по мнению этого гика, не должным образом, рассматривает это как покушение на свою работу, и примириться с этим очень трудно. Труднее всего понять, когда так реагирует другой такой же гик. Это просто удивительно. Потому что если даже они затвердят: «Хорошо, я не буду заходить в бухгалтерию и критиковать их неправильные таблицы», – они могут это понять. Но если речь о другом программисте – никакого понимания! «Почему это его нельзя критиковать? Что за ерунда!» И не только так, потому что это еще гуманный вариант, а просто: «Чего они такие чувствительные?»

Эндрю: **Итак, как помочь понять это обычному человеку? И как помочь приспособиться гику? Как помочь им взаимодействовать? Судя по всему, потенциальные проблемы в общении могут возникать с обеих сторон, и везде требуется понимание.**

Энди: Во-первых, гик должен осознать, что уважать других выгодно.

Эндрю: **По-вашему, это просто!**

Энди: Именно! Забавно, но недавно в OSCON было совещание с участием Киррили Роберт – она ведет веб-сайт *GeekEtiquette.com*, – я тоже присутствовал. Она выступала с введением в этикет – говорила, что этикет – это не вилки для салата и не свадебные приглашения. Это просто смазка для механизма человеческих взаимоотношений.

Она напомнила о принципе 80/20: три вещи, которые нужно помнить об этикете, – это уважение к другим, внимание к тому, что они говорят, и ежедневный душ. Если получается их выполнять, то это 80% успеха. Если нет, то это причина для людей не желать работать или общаться с вами.

И ведь ничего трудного. Она считает, что сам этикет – это вызов: да, порой действительно тяжело слушать, что говорят люди, если они вам надоедают. Да, порой сложно слушать других, не перестав их уважать.

Выход в том, что гик должен понять ценность уважения к другим и необходимость общественного взаимодействия. Потому что вопрос может стоять и так: «Если ты нахал, тебя уволят». Надо понять, что если хочешь делать то, что тебе хочется, для этого тоже нужна помощь других, а помогут тебе, только если ты такой человек, с которым хочется иметь дело. И раз уж вы здесь, то усвойте, что даже если вас не нервирует, когда кто-то ругает вашу таблицу, то другие могут относиться к этому иначе. Другие часто отличаются от вас. И различия между нами – между гиком и среднестатистическим Бобом – могут быть таковы, что вы ужаснетесь и скажете: «Боже, как кто-то может так думать?» Что ж, не надо пытаться понять, как и почему. Просто примите это к сведению. И уважайте образ мыслей другого.

Эндрю: **Круто. Отличный совет. Хорошо бы я услышал его раньше. Но как насчет Боба? На что похожа жизнь с его точки зрения, каково ему иметь дело с гиками? Как это отразится на команде?**

Энди: Бобу придется хуже, потому что Боб более нормален (то есть ближе к статистической норме). У вас в компании может работать 100 человек, и только пятеро из них – гики, фрики, кто угодно. И Боб имеет серьезные основания чувствовать себя ущемленным, потому что общественная норма предписывает ему: «Да, так себя и ведут гики, и придется с этим как-то мириться». Но Бобу приходится уважать и то, что

взгляд гика на мир тоже имеет право на существование. Гик смотрит на мир, желая улучшить его просто ради его улучшения. Он хочет улучшить технику вокруг себя, чтобы улучшить человека. И Бобу тоже предстоит это понять. Он может думать и по-другому, считая, что гик хочет не улучшить вещь, а наоборот, испортить ее. Но Бобу следует знать, что побуждения у гика самые лучшие. Действительно, гик просто хочет, чтобы все стало лучше.

Эндрю: **Сделаем шаг назад. Теперь у нас Боб с гиком не один на один. Вернемся к командам программистов. Надо создать свою команду. Допустим, вы в команде, и вам нужна помощь остальных. У вас начались проблемы. Вы можете быть лидером команды или ее рядовым членом, который понимает: «Надо же, это сильно влияет на меня и на всех вокруг». Какой следующий шаг? Что делать дальше? Как использовать свои знания, чтобы помочь программе стать лучше?**

Энди: Что ж, вы собираетесь не то чтобы сделать свою программу лучше, но по крайней мере создать ее быстрее. Потому что, за исключением пожара на сервере, нет ничего более вредного для производительности, чем команда, члены которой ненавидят друг друга или просто не способны работать вместе, или постоянно устраивают разборки с менеджером. Мне как менеджеру это противно. Это самая пустая трата времени и сил из возможных.

Я могу распознать проблему сервера и решить ее. Но если мои люди дуются друг на друга, то даже если я решил проблему в данный момент, это не значит, что она исчезла насовсем. Таким образом, в моих интересах как менеджера и руководителя команды сделать так, чтобы этого не случалось. Если назревает конфликт между людьми, в нем надо немедленно разобраться.

Дженни: **Насколько, по вашему мнению, конфликты влияют на поведение команды? В общем и целом, насколько конфликты были проблемой для вас при разработке ПО? Как вы считаете, закономерно ли, что два члена коллектива начинают друг друга недолюбливать и вам приходится вмешиваться?**

Энди: Большая разница между Бобом и гиком в том, как они рассматривают уважение. В обществе по умолчанию принято оказывать каждому определенную дозу уважения. Это закон. Вы говорите: «Вот человек, и я не собираюсь вести себя с ним как нахал, потому что это просто другой человек». Гик же считает, что уважение нужно завоевать. Как и неуважение – если кто-то поступает глупо, многие гики считают это достойным презрения.

В одной команде мне пришлось сказать: «В конце концов, можем мы все хоть немного уважать друг друга?» И кто-то спросил меня вполне

серьезно: «А что если не все тут достойны уважения?» Для меня как человека это был шок, что тоже бывает нечасто. Минимальное уважение – это то, чего многим не хватает.

Дженни: **Наверное, злобная была группа.**

Энди: Такими могут быть многие.

Эндрю: **Но это ведь дает и возможность «потайного хода» гику, который хочет улучшить с кем-то рабочие отношения. Если ему нужно работать с Бобом и он знает, что его заработок зависит от успеха этой работы, то он выкажет Бобу свое уважение, даже если тот его еще не заслужил. Ему отплатят взаимностью, и работать с ним будет куда проще.**

Энди: Да. А насчет неуважения – что ж, посмотрите на культуру поведения в Интернете. Почему возникает флейм? Потому что собеседнику совсем не нужно мое уважение. Я могу демонстрировать сколько угодно неуважения, потому что собеседник (если о нем вообще думать) – тот человек, который все это читает, – не нуждается в моем уважении. Это даже не сознательное решение: «Я не уважаю этого человека». Уважение тут и не ночевало. Гик даже не задумывается над такими вещами.

Эндрю: **Итак, допустим, вы член команды, и вам нужно сделать свою работу. Вы прочитали это интервью и думаете: «Черт, это про меня. И про моих коллег. У нас с Бобом нелады». Какой первый шаг легче всего сделать?**

Энди: Вот именно. Мы говорим здесь о поведении. Если у меня проблема с Джо из моей команды и он меня не уважает, то это не проблема. Проблема заключается в поведении. Если Джо агрессивен и враждебен – вот это уже проблема. Нужно сказать: «Смотри, Джо. Ты и я должны работать над этим проектом, и чтобы это состоялось, мне нужно, чтобы было так, так и так. Мне нужно, чтобы при проверке моего кода я не слышал нападок на него». Только *я*, *я*, *я* или *это нужно мне*. Не надо говорить: «Джо, ты нахал, потому что ругаешь мой код и не уважаешь меня». Нельзя заставить Джо вас уважать.

Если вы кого-то в чем-то обвиняете, говоря: «Ты делаешь то, ты делаешь это», – то вы открываетесь для ссоры. Если вы говорите: «Джо, ты меня не уважаешь», – то он немедленно отвечает: «Вовсе нет!» Можно по этому поводу спорить весь день, иногда так и бывает, уважает вас этот Джо на самом деле или нет.

Но вместо этого можно сказать: «Мне нужно вот что. Мне нужно, чтобы вся дискуссия касалась только этого кода. Так я и буду себя вести». Потому что спорить о поведении нельзя. Можно спорить о чувствах, отношениях или намерениях. А если мы что-то и знаем о гиках, так это то,

что они любят спорить, вести дебаты. Не попадайтесь в эту ловушку, не давайте себя вовлечь в спор о том, что Джо нахал, вас не уважает и так далее. Каждый раз, когда заговариваете о чем-то подобном, строго придерживайтесь темы поведения.

Еще одно: помните, что вашему менеджеру тоже придется иметь с этим дело или хотя бы учитывать ваши отношения. Задача менеджера – устранять помехи в вашей работе. Если Джо враждебен и неуважителен по отношению к вам или другим членам команды, то для работы это помеха. Что вы делаете? Вы сообщаете об этом препятствии, совершенно как если бы у меня сломался компьютер и я не смог бы ничего сделать за день. Оба случая ведут к снижению производительности коллектива.

И если ваш босс этого не понимает, если он не осознает таких тонкостей командной динамики, то ваша команда, по всей видимости, обречена, так что лучше подыскать себе другое место работы.

Дженни: **Я бы хотела отвлечься от этикета гиков и команд и перевести разговор на более высокий уровень – поговорить об управлении командами и работе с людьми в командах.**

Энди: Во-первых, вы как руководитель команды или менеджер должны понять, что команда должна работать вместе. Проблемы этих людей – это ваши главные потенциальные препятствия, и все, что мешает им работать вместе, способно развалить весь ваш проект. Нужно осознать, что в вашей команде могут оказаться люди, которые с ростом производительности станут отрицательным фактором. Если у вас в команде человек пишет по сто строк кода в день, но снижает продуктивность всех остальных членов коллектива, если никто не хочет с ним работать, то это значит, что время, которое раньше тратилось на написание программ, теперь уйдет на попытки всех примирить, менеджер должен это понимать.

Таким образом, нужно понять, что те, кто вредит коллективу, должны немедленно изменить свое поведение. Иначе весь проект отправится в сортир. И если человек не осознает, что работа команды важнее, чем его индивидуальный вклад, то такой человек вам не подходит и от него надо избавиться. Значит, вывести из команды, из компании – отовсюду.

Эндрю: **А на стадии создания команды? Как избежать этого сразу же?**

Энди: Если вы говорите о своей команде, то человеческие качества надо учитывать уже при найме. Обычно, когда я кого-то нанимаю, то провожу два круга собеседований. Первый – насколько претендент приемлем для меня, это касается основных технических навыков. Вы умеете писать код? Насколько вы владеете языком программирования X? Знаете

ли вы теорию баз данных? И все в таком духе. Это нетрудно. И не отнимает много времени, потому что нас двое – я и претендент на вакансию. И если человек преодолевает начальный барьер «Может ли он писать код? Может ли он выполнять работу?», то наступает второй этап: «Может ли он работать в команде и подходит ли он этой команде?»

«Подходит» – в очень узком смысле, поскольку все команды разные. Все отделы разные, все организации. И как и во всех остальных вопросах общения между людьми, понятия хорошего и плохого здесь неприменимы. Нет ничего плохого в том, что человек хочет сидеть в своем отсеке и ни с кем не разговаривать по восемь часов в день. Здесь нет ничего плохого, просто вы не подходите моей команде. Если вы захотели в ней работать, то это был плохой выбор. Это плохой выбор для тех, кто хочет работать у меня, потому что мои команды работают по-другому.

Таким образом, второй, а иногда и третий круг собеседований – это сбор всей команды и разговор с претендентом. Возникают и технические вопросы. Но главным образом я отхожу в сторонку и смотрю, как он общается с остальными, слушаю, о чем они разговаривают. Остальные члены команды задают вопросы. Обычно направляют разговор главные разработчики. Они подходят поговорить с человеком и посмотреть, как он вписывается в нашу команду. И вы говорите, например, о конфликтах: «Расскажите о каком-нибудь конфликте с вашим коллегой, когда вы думали, что ваш оппонент – просто полное дерьмо. Как вы с этим справились?» Или: «Расскажите о каком-нибудь случае, когда кто-то из команды обошелся с вами, по вашему мнению, несправедливо». «Расскажите о каком-нибудь трудном члене вашего коллектива: как вам удавалось с ним работать?»

Это не все собеседование, но важнейшая его часть. Просто послушайте несколько таких историй (это должны быть именно истории, а не диалоги типа «Вы хорошо ладите с другими членами команды?» – «Конечно, да!» – так любой ответит), и вы поймете, сможет ли человек вписаться в ваш коллектив.

Эндрю: **Итак, второе собеседование проходит не только с вами, но и со всеми членами коллектива, правильно?**

Энди: Да, на втором собеседовании обычно присутствуют старшие разработчики, я и даже, возможно, мой босс. Будут обсуждаться технические стороны, но подлинная наша цель – это взаимодействие. Я говорю людям, что очень требователен. У меня одна вакансия, и я очень требователен к претендентам на эту вакансию, потому что люблю эту команду, люблю свою работу и не позволю кому попало затесаться в мою группу. Есть, конечно, и более практические аспекты. Во-первых, всегда болезненно кого-то увольнять.

Эндрю: **А как насчет недостатков подобного подхода? У всех ли есть право вето? Допустим, всем понравился кандидат, кроме одного человека?**

Энди: Это вовсе не голосование. И тем более не опрос с двумя вариантами ответа. Я прошу только высказать свое мнение. Я говорю: «Те, кто за столом, покажите мне большой палец». И мне везло в том отношении, что никогда не было случая, чтобы кто-то расходился во мнениях с остальными. Никогда не бывало так, чтобы один говорил: «Отличный парень, давайте возьмем», – а остальные утверждали, что он придурок, и наоборот. Потому что я считаю, что если достаточно долго поговорить с человеком, выяснится, подходит нам такая личность или нет.

И опять же – насчет слова «подходит»: «Действительно ли он подойдет всем в нашей команде?» Дело в том, что тот недостаток, которого я действительно страшусь, таков: я не хочу, чтобы команда обязательно была единой массой. Я не хочу, чтобы думали, что мы берем на работу только таких же, как мы, потому что если брать только себе подобных, то не придумаешь ничего нового. А я стараюсь искать новое.

Дженни: **Вы использовали слова *подходит* и *личность*. Скажите, не склонны ли мы брать на работу того, с кем хотелось бы дружить или тусоваться, вместо того чтобы реально испытывать профпригодность кандидата? Что это дает команде? И вообще как-то на нее влияет?**

Энди: Я с таким не сталкивался, и мне никто ничего подобного не говорил. Никто не говорил: «Я бы принял этого парня, потому что думаю, что с ним было бы неплохо побухать». У нас в команде достаточно различий, не все одинаковы, так что я об этом особо не забочусь. Думаю, каждый, кто приходит на собеседование от команды, понимает важность приема на работу человека, если с ним потом придется ежедневно иметь дело. Мы ведь нанимаем людей, с которыми проводим больше часов бодрствования, чем со своими супругами.

Эндрю: **Как вы думаете, должна ли хорошая команда обязательно состоять из тех, кто любит собираться и после работы?**

Энди: Нет, безусловно, нет! Даже не считаю это условием. Я работал в командах, где были люди, с которыми я бы вообще никуда не пошел, потому что в этом смысле они были мне неинтересны. Может, они любят играть в пляжный волейбол, а мне на него наплевать. Один наш парень пару раз в неделю после работы бежит играть в гольф. А по мне, так это невероятно скучно. Но тут все в порядке. Опять же, не все, как я, в свободное время работают над проектами с открытым исходным кодом. Не все, как Майк, играют на гитаре в традиционной ирландской группе (я даже толком не представляю, что это такое). У всех есть свои

сторонние интересы, и это нормально – вовсе не обязательно, чтобы они совпадали. Мы ведь и так собираемся на восемь или больше часов в день и вместе работаем.

Надо сказать, что с некоторыми коллегами я дружу. И это тоже естественно. Но вовсе не обязательно для каждого.

Эндрю: **А как насчет ваших проектов с открытым исходным кодом? В какой мере это относится к ним?**

Энди: Деятельность людей, которые общаются друг с другом в ходе проекта с открытым исходным кодом, во многом похожа на деятельность людей в компании. Разница в том, что в проекте с открытым кодом нет никого, кто мог бы сказать: «Приспособься и будь как все, а не то вылетишь».

Эндрю: **Как вы думаете, что привносит отсутствие такого человека в динамику проектов с открытым кодом? Это помогает или вредит?**

Энди: Что ж, одна из отличительных черт проектов с открытым кодом в том, что руководители в них становятся руководителями. Посмотрите на Perl, в котором работаю я. Да, Ларри Уолл – лидер Perl, потому что это его язык, он написал и изобрел его. Но кроме него, все другие руководители определились по тому, что они сделали. Если вы работаете над этим и этим, от сих до сих, пишете эти модули, и эти модули всем нравятся, и вы помогаете другим, то тогда вы зарабатываете уважение, и вас начинают рассматривать как руководителя. А вовсе не так, что «Ларри назначил того-то и того-то главными в таких-то группах».

Думаю, такой подход удовлетворяет мечты гиков о меритократии: те, кто делает больше и лучше всех, получают за это вознаграждение. И если вознаграждение это состоит в том, что человека рассматривают как уважаемого члена общества, то это только естественно.

6

Как воодушевлять людей

Интервью с Кеоки Андрусом

Кеоки Андрус – специалист по улучшению совместной работы команд. Мы узнали о нем, услышав его лекцию о производительности и потенциале, когда он возглавлял операционный отдел в Intuit. У него были отличные идеи по поводу того, как превратить несколько туманные понятия, такие как «видение» и «потенциал», в практичные, работающие технологии улучшения производительности команд. Эндрю встретился с Кеоки, чтобы поговорить об этом.

Эндрю: **Как вы стали специалистом по командам разработчиков?**

Кеоки: Так я в программировании уже 20 лет. Начинал в Microsoft, где работал над первыми версиями Word. Работал с одним из менеджеров проекта Office, когда мы превращали его в программный пакет. Затем долго работал руководителем в Novell, занимался стратегическим менеджментом продуктов и проектов.

Мы с женой часто шутим на эту тему. Каждый раз мы начинаем обсуждение с вопроса: «Что нам делать с нашим товаром?» Обычно несколько дней спустя это перерастает в дискуссию: «Как управлять компанией? Мои люди недовольны. Как решить эти проблемы?» И каждый раз это происходит из-за того, как работает компания, из-за связности организации – эти вещи очень важны для успеха.

Эндрю: **Когда вы всерьез набили руку в обеспечении работоспособности компаний?**

Кеоки: Некоторое время я уже работал в Intuit. Я управлял компанией, и меня срочно перебросили в другое место. В Ореме (Юта) они открылись за шесть лет до того, а сейчас филиалу грозил крах. Поговорив кое с кем в Intuit, я понял, что они уже практически собрались прикрывать филиал из-за непроизводительности. «Настроение у сотрудников на нуле, и если что-то с этим не сделать, то придется закрыть. Так что вы наш последний шанс. Почему бы вам не попробовать?» Ну, я поработал там, и мы смогли из забитой, малопроизводительной, разрозненной организации стать одной из самых продуктивных групп инженеров во всей компании. Мы удвоили размер предприятия. Кто-то получил повышение, а кому-то пришлось уйти, но в целом успех был значительным.

Эндрю: **Да, впечатляюще. И как же вы этого добились?**

Кеоки: Прежде всего я отдельно поговорил с каждым сотрудником и попытался понять, что они сами думают о текущих трудностях.

Я преследовал здесь две цели. Во-первых, действительно хотел понять их и то, что тут, по их мнению, происходит. Во-вторых, хотел, чтобы они поняли, что я действительно забочусь о них. Поскольку моя главная гипотеза такова: если люди не верят, что тебе это действительно нужно, им наплевать на то, что ты говоришь.

Скоро я обнаружил, что есть много недовольных. В команде было несколько сотрудников с очень высоким потенциалом, которые могли работать намного производительнее, но были подавлены. Им требовалось человеческое участие, хорошие рабочие отношения, а их одного за другим отрывали от проектов, заставляя работать удаленно.

Что ж, у многих это очень хорошо получается. Но для тех, кому изо дня в день нужно межличностное общение с коллегами, это действительно

нелегко. Я отметил этих людей; они были лидерами филиала и к ним относились с уважением, но чего-то все же не хватало.

Сначала я пытался сказать им: «Придется это прекратить», – но остался непонятым. Затем я заметил, что один из менеджеров пользовался здесь всеобщим уважением. Собственно, даже не уважением. Его здесь просто любили.

И я тоже стал относиться к нему так, и до сих пор я ценю его очень высоко. Это такой потрясающе компанейский парень, но высшее руководство пинало его за то, что его подчиненные плохо работали. Я отозвал в сторонку лучшего из этих высокопотенциальных плохих работников и сказал: «Послушай. Ты можешь быть суперзвездой, просто тебя расстраивает то, как здесь идут дела». Он согласился. И я добавил: «А ты знаешь, что твои действия вредят вашему менеджеру?»

Это потрясло его, потому что он любил того парня и вовсе не хотел причинить ему вред. Я посидел с ним, и мы обсудили его дальнейшие карьерные возможности и нынешнюю ситуацию.

К тому времени он уже понял, что я неравнодушен к его судьбе, поскольку я не осудил его, даже когда он весьма резко объяснил свое недовольство.

Я предложил менеджерам, которые руководили удаленным проектом, перевести его обратно в местную команду. Они были рады избавиться от недовольного. А он увидел, что я готов сражаться за него, и когда мы к этому пришли, то сели и обсудили одну за другой все те сильные стороны, которые я в нем видел.

Я сказал: «Знаешь, когда ты сможешь применить их все, то обязательно станешь суперзвездой, но для этого надо стать занудой, а это не по тебе. Может, просто попридержать себя немного? Я скажу тебе то, чего не скажут другие. Ты остроумен и любишь это показывать. Но нельзя ли не всегда делиться этим с остальными? Можно не шутить, когда идет планерка?» И он все понял!

Теперь нужно было разобраться с менеджером. Чтобы он понял, как управлять сотрудниками. Вице-президент сказал мне о нем: «Пожалуй, этого парня лучше уволить. У него большой потенциал, но сейчас он только мешает, и мы не знаем, что сделать, чтобы он работал нормально».

Я встретился с этим парнем и обсудил с ним всю его жизнь. Потом я пошел к вице-президенту и сказал: «Вы знаете, он – суперзвезда. С маркетингом у него не очень хорошо, но он – суперзвезда. Ему только нужно кое-что изменить в своей манере общения с другими». Я поговорил с менеджером так же, как беседовал с разработчиком. Сказал ему примерно следующее: «Да, ты классный, народ за тебя горой. Здесь все хорошо,

но давай укрепим пару других позиций, и ты станешь именно тем, кем надо».

И он начал работать над этим. Спустя полгода программный пакет был уже почти готов.

Эндрю: **Значит, начинали вы всегда с хорошей командой, где присутствовали все необходимые технические навыки, но что-то мешало — то, о чем программисты обычно не думают. И эту проблему вы решали, ломая коммуникационные барьеры: помогая людям лучше общаться, подчеркивая их сильные стороны и приглушая слабые. Так? Общение — всему голова?**

Кеоки: Я предпочитаю разговаривать один на один, поэтому разговоры тет-а-тет – моя сильная сторона. Беседовать по телефону мне нравится меньше. С этим менеджером получилось забавно, потому что сначала вице-президент сказал: «Почему вы его не уволили? Этот парень неудачник. Там нужен новый менеджер». А я ответил: «Видите ли, я действительно считаю, что у него есть замечательные качества и способности, надо только помочь ему их проявить». На что вице-президент сказал: «Впервые за шесть лет я слышу о нем хоть что-то положительное. Вы уверены?»

В тот же год я получил отзыв о своей работе, и он был очень неплох. Но там, например, говорилось: «Просто поразительно, чего вы смогли добиться, учитывая что вам пришлось работать с такими людьми, как этот парень и другие, не более талантливые». Это было как оскорбление, потому что сам я считал их отличными работниками.

И каждый раз, когда им удавалось что-то хорошее, я старался убедиться, что это действительно хорошо. И не только из соображений маркетинга. Эти ребята были просто золото, но об этом никто не знал. И после некоторого успеха я настоял, чтобы им больше доверяли. Скоро о том же самом менеджере все стали говорить: «Да он просто феномен!»

На следующий год стало еще забавнее. В моем отзыве было сказано нечто поразительное: «Да, ваш филиал работал фантастически. Но чего ожидать, если им руководит такой великолепный менеджер?» Мне и добавить было нечего! Потому что именно этого я для него и добивался. Но все же они меня насмешили.

Эндрю: **Итак, какой опыт вы извлекли из всего этого? Какие-нибудь ключевые принципы?**

Кеоки: Что я понял, так это то, что вначале у этих людей не было правильной цели. Они смотрели на себя как на винтики в рабочем механизме. Никто не видел собственной крутизны, своего потенциала. И некоторым я в этом помог.

Первым делом было осознать, что это будет величайшая инженерная группа в компании, что на нас будут смотреть как на авангард, как на законодателей мод, как на тех, кто всегда впереди, кто делает что-то новаторское. Что мы станем теми, кому доверяют самые сложные задачи.

А до того они ничем не отличались от дворников или мусорщиков: «Кто-то должен написать этот код. Так почему бы не они». И я решил: «Покончим с этим». Я сразу заметил, что каждый инженер в этом филиале действительно был талантлив. Проблема, собственно, была в том, что у каждого из них был свой подход к разработке.

Каждый из них все делал – в том числе писал код – по-своему. Все они делали это хорошо и достаточно понимали друг друга, чтобы один мог разобраться в работе другого и что-то поменять на ходу. Но если в группу приходил новичок, это было для них ударом. То есть у организации не было каких бы то ни было возможностей роста.

Если вы старожил, то все отлично. А если нет, то ужиться с ними было невозможно. С посторонней точки зрения, работая с ними, нельзя было найти общий язык, даже будучи программистом. И я сколотил группу, и мы начали такой жесткий процесс обновления, что все только скулили.

Эндрю: **У меня был такой же опыт. Наверное, вы столкнулись с внутренним сопротивлением команды, потому что они не хотели менять свои методы создания программ.**

Кеоки: Все они говорили что-то вроде: «Ужас, как трудно». Я отвечал: «Знаете что, ребята? Мы на пороге подъема. Мы собираемся создать организацию, которая отличается наивысшим качеством, и вы можете это. Да, это очень трудно». Я знал, что столкнусь с неприятием, что будет бунт. Но я знал и то, что нет такого процесса, который устроил бы всех, и что любой процесс порой может казаться глупостью.

У меня есть жизненный прицип. Об этом знают все мои коллеги; он написан на стене: «Если сюда вошла глупость, то ты имеешь моральное право убить ее, кто бы ее ни сопровождал».

Эндрю: **И люди восприняли эту идею? Даже когда вы объявили их мысли глупостью?**

Кеоки: Те, кто со мной работает, хорошо это знают. Даже если глупость входит со мной, убейте ее. У меня никогда не было с этим проблем. Я посмеюсь, если вы укажете мне, какой я идиот. Никакого эгоизма тут нет.

Единственное, чего я хотел от них, – просто дать шанс новым представлениям о работе, потому что каждый сопротивлялся переменам. Сначала я был тверд: «Мы сделаем все, что положено, какой бы глупостью это ни казалось». В ответ: «Угу». Но в то время они мне уже немного доверяли, потому что сам филиал начал пользоваться доверием, все шло

вроде бы хорошо, с ними стали нормально обращаться, им нравилась их работа, и они думали: «Что ж, все это сделал Кеоки. Может, он и тут прав, хоть и выглядит это чертовски глупо».

Так что они мне достаточно доверяли. Я сказал: «Не спешите, ребята. Давайте начнем, а потом, если увидим, что это глупо, то убедимся, что это глупо, а не просто будем думать, что это глупо, – только тогда мы передумаем». Я заставил их поработать так несколько месяцев. Пару раз они говорили: «Нет, это не для меня», или: «Я знаю, что в MIT так считают, но это же чушь».

Я отвечал: «Ладно, давайте поговорим, почему это чушь. Это чушь, потому что вам это не нравится, или же это чушь, потому что это не подходит под наши условия? Если вы сможете объяснить мне, почему это не подходит для нас, для нашей системы, то мы все поменяем». Мы внедрили несколько изменений, что действительно помогло организации.

У нас все стало получаться лучше. В то же время они стали понимать, что я живу по своему принципу и не заставлю их делать глупости без особого повода. Пройдя через эту фазу, мы начали давать результат.

По сравнению с остальными командами разработчиков мы успевали вовремя. И качество нашего труда было потрясающим: на пару уровней выше, чем у остальных.

Эндрю: **Действительно, впечатляет. Но мне кажется, вы ничего поразительного и не делали: в основном ваши действия продиктованы здравым смыслом. Итак, вы сказали, что заставили людей поверить в себя. Что было дальше?**

Кеоки: Дальше они должны были начать думать: «А что, нам надо быть лучшими». Ну, а что делают великие инженеры? Они создают новое. И мы начали программу поощрения нестандартных решений, чтобы понять, какие инновации мы можем придумать.

Мы начали получать патенты. В первый год реализации этой программы, ориентированной на инновации в области создания интеллектуальной собственности, мы составляли всего полпроцента инженеров компании, а патентов у нас было процентов семь или восемь от всех инноваций организации.

На следующий год это число возросло до 16–17%.

Эндрю: **Какой скачок. Как вам это удалось? Ввели еженедельные сеансы мозгового штурма?**

Кеоки: Прежде всего, я почти не подвергал критике идеи, которые выдвигали сотрудники. Мне нужны были идеи, они должны были течь по-

током, а не теряться под критикой. А потом мы представляли аналитикам некоторые идеи и говорили, что заинтересованы в них.

И это было забавно: однажды я получил из юридического отдела письмо, где говорилось: «Знаете, надо бы обсудить, какие ваши патентные идеи стоит развивать, а какие нет». Но кто-то им сказал: «Да ладно вам! Это единственный человек, который делает то, что от него действительно требуется». После чего следующее электронное письмо гласило: «Все в порядке, разговора не требуется».

Эндрю: **Все же я бы хотел узнать, как вы достигли такого уровня инноваций, потому что порой новшества кажутся мне почти мистическими. Откуда берутся идеи? Понимаю, что мы хотели поговорить о командах, но мне кажется, что эти два предмета имеют друг к другу непосредственное отношение. Как заставить людей в команде выдавать хорошие идеи? Как улучшить работу, как помочь сотрудникам-новаторам в группах?**

Кеоки: Ключ ко всему – лидерство. Прежде всего надо определить цель, в которую каждый бы свято верил, – это включает в работу. Второе – собственно, как. Если у людей есть цель, то у них появляется энтузиазм, который, в свою очередь, побуждает к новаторству.

На собраниях команды я не продвигаю собственные идеи, а внедряю их в группу. Мы обсуждаем их до тех пор, пока все не загораются. Я вбросил идею с патентами, и им всем понравилось, потому что это был шаг к величию и всемирной известности команды.

После этого на каждом собрании кто-то выступал с идеей и спрашивал: «А это можно запатентовать?» И ему отвечали: «Погоди-ка; да, наверное, можно». Вскоре, проходя по офису, я увидел, что у одного парня на доске были записаны все идеи, которые родились в группе.

Обсуждение возможных инноваций вошло у нас в обычай. Все волновались, когда всплывал вопрос: «Это можно запатентовать?» – и этим надо было пользоваться. Все это было частью той цели, которую избрала себе команда. Если вы определили цель, то нужно ей следовать, и если все это ощущают, горят энтузиазмом, то творческий импульс возникает спонтанно.

Кроме того, я хотел придать им уверенности. Например, один инженер, блестящий специалист по базам данных, никогда не пытался сделать что-то сверх своих обязанностей. Как многие инженеры, он говорил: «Я не обещаю того, чего не могу сделать». Больше года мне пришлось объяснять этим парням, что я не съем их, если они проявят инициативу, сделают что-то выходящее за рамки их штатного расписания.

Будьте храбрыми. Это правило. Узнавая что-то новое, мы умнеем. Мы знаем, что научились чему-то новому, и это меняет наш образ мыслей. Только кретин ругает подчиненного, который не знает того, чему ему только предстоит научиться.

Однако инженеров всегда наказывают, поскольку для них есть штатное расписание. Но я в эти игры не играю. И мои инженеры начали это понимать. Тот сотрудник никогда не обещал то, чего не сможет выполнить. Поняв это, я сказал себе: «Он мой».

Однажды он пришел ко мне в офис и сказал: «Знаете, мы не можем решить эту задачу. Здесь ничего нельзя сделать». А я посмотрел на него и ответил: «Джефф, ты один из самых блестящих инженеров, каких я знал в жизни. И только что один из величайших инженеров всех времен сказал мне: „Здесь ничего нельзя сделать“».

Иди-ка к себе, Джефф, еще подумай, а к трем часа дня возвращайся и скажи мне, в чем загвоздка». Он посмотрел на меня и сказал: «Но я же говорю, это невозможно».

«Да, было бы невозможно – для обычного инженера, но ты-то не обычный инженер! Ты один из лучших». Только нужно говорить это искренне. Если у вас нет близких, личных отношений с человеком, когда он воспринимает вас всерьез, то он может просто отмахнуться.

Эндрю: **Но с тем инженером у вас были именно такие отношения, он доверял вам и поверил, когда вы это ему сказали?**

Кеоки: Да. Он знал, что я его уважаю. Я часто проделывал это с ним – именно с ним, потому что к каждому нужен индивидуальный подход. Он говорил: «Ненавижу, когда ты так делаешь».

И каждый раз через пару часов он вламывался в мой офис с криком: «Получилось!» – или же я проходил мимо его отсека, а он звал меня: «Эй, сюда, сюда!» Потому что он был действительно гением, и мне оставалось просто поставить его в такое положение, которое он рассматривал как личный вызов, когда он чувствовал личную мотивацию.

Это срабатывало, потому что он знал, что я его не съем. Потому что я говорил: «Если ты не сможешь понять, в чем тут дело, то и никто не сможет, но я думаю, что у тебя получится». И он никогда меня не подводил.

Эндрю: **Да, похоже, вам попалась на самом деле сильная команда инженеров. У вас были блестящие таланты и команда, которая могла работать очень хорошо, но ей мешали личные проблемы, трудности в общении и отсутствие цели. Видимо, главное ваше достижение было – понять, чем мотивировать каждого сотрудника, и дать им это.**

Кеоки: Был у меня другой инженер – надеюсь еще когда-нибудь с ним поработать. Он просто бог. Он заканчивает свою работу раньше срока и с отличным качеством, затем помогает остальным членам команды решать их задачи, а потом идет еще дальше, занимаясь задачами тех, кто не входит в команду.

Этот парень очень любит играть. Он не заводится, пока не придет в игровое настроение. Очень любит настольный футбол и другие игры. И если кто-то говорит: «Когда у тебя нерабочее настроение, ты не сделаешь ничего путного», – я отвечаю: «Бред!» Нужно просто понять, что подстегивает творческий импульс человека, так что он зажигается энтузиазмом.

Ах, как он любил настольный футбол! И у нас на работе был этот автомат. В него играют по двое с каждой стороны. Если хотят сыграть, ходят по офису и кричат: «Три!» – значит, кто-то уже занял одно место. Второй отвечает: «Два!» – то есть осталось найти двоих, еще кто-то кричит: «Один!», затем слышится: «Я!» – и вот их уже четверо.

Сыграв, они возвращались на свои рабочие места. Скажете, что это глупо: «Поощрять бездельников!» Но я заметил, что многие, оторвавшись на 10 минут от стола ради игры, возвращаются к нему с такой кипучей творческой энергией, что задача решается мгновенно.

Однажды я пригласил к себе Джейсона, того отличного инженера, и спросил его: «Джейсон, расскажи мне, как работает твой мозг. Ты все время играешь в игры, но при этом ты самый продуктивный инженер, которого я видел за всю карьеру. Есть ли связь между игрой и таким потрясающим уровнем качества работы?»

И он ответил: «Мне просто нужно прочистить мозги. Допустим, есть у меня какая-то задача. Я отойду, поиграю немного и уже могу ее решить». А я сказал: «Отлично, Джейсон. Не знаю, скажу ли я это кому-то еще, но как только тебе захочется поиграть, требую, чтобы ты играл, потому что если это помогает тебе работать на таком уровне, то тебе нужно играть столько, сколько ты захочешь».

И он меня никогда не подводил. Вот, в общем-то, и ответ на ваш вопрос о творческом импульсе. Надо понять, что подстегивает каждого сотрудника, и стараться помочь им пребывать в таком состоянии.

Эндрю: **Удивительно, вы постоянно говорите об *уважении*, *энтузиазме*, *творческом импульсе*, даже о *любви*. Говоря о мотивации, многие упоминают уважение и творческий импульс, но при этом редко услышишь об эмоциях. Думаю, вы используете уважение и любовь как орудия управления, как инженер технические инструменты. Это вроде *сленга*, в хорошем смысле. Говоря «любовь», вы вряд ли имеете в виду слабые вздохи с телеканала Hallmark. Вы вкладываете сюда какое-**

то техническое значение, получая важный инструмент для сплочения команды. Как вы к этому пришли?

Кеоки: Прежде всего, я еще на заре своей карьеры обнаружил, что наблюдателен. Инженеры чаще других думают о себе в стиле доктора Спока: «У меня есть логика. И это все».

И я начал следить за тем, как инженеры реагируют на критику их кода. Порой я думал: «Да это двухлетний ребенок. Этот парень невероятно эмоционален». И чем больше я наблюдал за инженерами, тем больше понимал, что они так же эмоциональны – если не больше – как и все остальные, и что эмоции это большая часть их работы. Они просто хорошо структурируют свое общение.

Я понял, что им нужны эмоциональные отдушины и взаимоотношения, как и всем остальным. И это было первое, что я осознал.

Эндрю: **Интересно, насколько соотносится с уже сказанным мой следующий вопрос. Как у вас появляются идеи в свете общей цели? Видимо, цель позволяет вам реально повысить производительность, таким образом улучшить работу, что вдохновляются и те, кто иначе бы не стали об этом говорить.**

Мне кажется, что «цель» – еще одно слово, которым вряд ли пользуются инженеры и разработчики. Оно скорее из сленга бизнесменов и менеджеров. Может, и нет, но раньше я, во всяком случае, такого не слышал.

Я и сам не сразу принял идею цели для проекта или команды. Я понимаю, что цель связана с тем, для кого вы делаете проект, с нуждами клиента, с пониманием того, какую задачу вы решаете. Но слушая вас, я подумал, что, пожалуй, должно быть какое-то более интуитивное определение цели. Не могли бы вы объяснить смысл цели мне, а заодно и тем, кто хотел бы выступить с интересными идеями или внести улучшения в работу над проектами?

Кеоки: Думаю, смогу. С целью обычно дело плохо.

Например, ее формулируют так: «Наша цель – стать главным поставщиком телекоммуникационного оборудования на рынке». Или: «Наша цель – перейти с производства штыревых антенн на что-то другое». Такие цели бесплодны.

Если вы хотите, чтобы люди работали лучше, надо надавить на их эмоции. Эмоция – это источник творческого импульса. Это источник инноваций, долгих часов работы и всего остального, чего вы хотите от сотрудников. Но это не значит, что надо приставить всем пистолет ко лбу. Или просто стоять над душой. Как же определить цель, чтобы вызвать эмоции?

Если люди не прониклись ею, то это плохая цель. Так что, рисуя цель, нужно быть отчасти проповедником.

Приведу свой пример цели. Думаю, цель должна быть всегда. Мои сыновья участвовали в шоу-хорах. Знаете, что это такое?

Эндрю: **Нет.**

Кеоки: Шоу-хор – это нечто вроде Бродвея для средней школы. Они поют и одновременно танцуют. Просто великолепно. Мой младший сын – чемпион страны, а старший был признан лучшим среди мальчиков. Мы собрались ехать в Брэнсон (Миссури) с его группой.

Я живу в Юте, и ехать надо 30 часов на автобусе. Можно было полететь, но предпочли самый дешевый вариант. Нас попросили: «Нужны родители-добровольцы, чтобы ехать с ребятами 30 часов в автобусе в обе стороны». Все родители согласились, думая при этом: «Что ж, надо так надо. Придется 30 часов трястись посреди усталости, эстрогена и тестостерона. Какой кошмар».

А я подумал и сказал: «Берусь за это. Но что мне делать? Просто сидеть по 30 часов туда и обратно, чувствуя себя унизительно лишним в компании подростков? Нет, мне нужна цель». Хорошо, но какая цель тут может быть?

Поразмыслив об этом, я сказал себе: «Постойте, но ведь эти ребята едут, чтобы воспользоваться одной из самых замечательных возможностей в их жизни: они будут раскрывать свой потенциал. Так, стоп. Ведь я – тот человек, который понимает в лидерстве и помогает людям раскрывать свой потенциал. Что если я тут пригожусь?»

Я подумал: ребята мало что знают об этом. А что, если моя цель – сделать эту поездку одним из самых блестящих моментов в их жизни, хотя бы для некоторых? Они вспоминали бы его и думали: «Да, тогда я впервые ощутил свой потенциал. Тогда я впервые ощутил величие момента».

Это изменило мои обязанности в поездке: теперь мне нужно было не присматривать за ребятами 30 часов туда и обратно и во время пребывания в гостинице. Я должен был пользоваться каждой минутой, чтобы помочь им ощутить собственную крутизну. Собственно это и была цель.

В результате благодаря цели мне было чем заняться каждую минуту поездки – во-первых, трястись в автобусе стало куда легче. Во-вторых, когда лидеры не знали, что делать, я смог каждого отвести в сторонку и рассказать ему кое-что о величии. В результате они действительно добились очень многого, и это был потрясающий опыт в их жизни. Я до сих пор дружу с некоторыми из тех подростков.

Я поделился этим с вами, чтобы привести пример того, какой простой и приземленной может быть цель. То есть она может быть связана

и с чем-то столь же обычным. Надеюсь, что слушая мою историю, вы прониклись и подумали: «Эх, если бы в том автобусе с ребятами был я!» И это – признак настоящей цели, которая заставляет думать: «Я хочу в этом участвовать». Даже в чем-то столь же скучном, как 30-часовая поездка на автобусе с подростками.

Эндрю: **Мне встречались проекты, заставляющие подумать: «Эх, вот бы попасть в ту команду!» Возможно, это как раз связано с целью. Как-то раз я смотрел интервью с разработчиками одной из моих любимых видеоигр и ощутил сходное чувство.**

Знаете, я уже писал – и не раз читал это у других, – что, запуская проект, хочешь, чтобы все тут же заработало. Хочешь избежать проблем на своем пути. И все – в том числе мы с Дженни в наших книгах – говорят, что главное – определить цель в самом начале проекта. И что нет лакмусовой бумажки, которая показала бы, что цель выбрана верно. Пожалуй, вы сейчас ответили на этот вопрос. Хороший индикатор того, что цель выбрана верно, – когда, услышав о ней, вы хотите попасть в этот проект.

Кеоки: Именно. То же самое верно и для команд, когда вы их подбираете. Я всегда получаю нужных мне людей, потому что, приглашая их на работу, я не завожу шарманку о том, что «ваши обязанности будут такими-то». В моей команде все находятся на пути куда-то вверх по карьерной лестнице, и этот путь уникален для каждого. Я говорю это всем.

Моя цель как лидера – подобрать каждому правильно место, на котором ему будет комфортно, и выработать цель, к которой он будет двигаться. Я помогаю ему на этом пути, если он хорошо работает в команде. Я учу его лидерству. Я учу его всему, чему он захочет научиться, чтобы он смог продвинуться туда, куда хочет.

Я привожу тем, кого приглашаю на работу, примеры своего обращения с людьми и говорю: «А теперь поговорите с членами моей команды. Можете спросить их обо всем, что хотите. Спросите любого моего сотрудника». Я даю им поговорить с теми, кто уже работает со мной, и любой из них скажет: «Да, он действительно заинтересован не только в моей работе, но и во мне как личности».

Претенденты всегда возвращаются и говорят: «Я никогда не сталкивался с подобной работой и отдал бы все, чтобы быть в такой команде». Так что я всегда побеждаю.

Эндрю: **Надо же, я занимался набором на работу много лет, а ничего подобного не делал. Скажу теперь, чего я хочу для своей команды, для своей компании. Я хочу, чтобы мои команды работали так же. Я вижу, как это будет выглядеть в итоге, но как этого достичь? Как сделать**

первый шаг к этому в моем положении? Вы рассказали, что начинали с командой, где был большой потенциал, и изменили в ней отношение к работе. Допустим, у меня есть команда, и я знаю, что у нее внушительный потенциал. Как изменить ее к лучшему? Как перейти из пункта А в пункт В?

Кеоки: Ну, во-первых, можно позвать меня, я приду и помогу им. Я этим всю жизнь занимаюсь, знаете ли.

Эндрю: **(Смеется.)**

Кеоки: И я люблю это делать. Процесс развития навыков – это процесс осознания, что тебе нужен этот навык, обучение ему и применение. А чтобы стать настоящим мастером, необходима помощь мастера.

В книге вы можете добавить к этому что хотите, но чтобы стать специалистом в чем-то, нужно учиться. А теперь хорошая новость. Людям всегда очень хочется, чтобы о них заботились. Подумайте об этом немного. Вы ведь хотите быть понятыми, так? Все хотят. Все хотят чувствовать свою значимость. Это универсально.

Если вы обратитесь к своей команде и скажете: «Да, я все сделаю. Мне не очень хочется, но я это сделаю», то все потеряно: босс из вас неважный.

Но если вы искренне желаете общаться с сотрудниками, попытаться узнать их поближе – кто они такие и что для них важно, облегчить им достижение цели или помочь определить цель, то они ответят на это в геометрической прогрессии, поскольку все любят, чтобы к ним относились с заботой и участием. Это жизненно необходимо всем.

Они страстно желают этого. Вспомните известный образ человека, умирающего в пустыне от жажды. Сколько он выложит за глоток воды? Так вот, почти на любой работе каждый хочет, чтобы к нему относились как к человеку.

Я никогда не встречал никого, кто не хотел бы быть понятым (разве что человек находился на грани психического расстройства). Кто-то говорит, что не хочет, но просто понаблюдайте за его поведением – и вы поймете, что это не так.

И я как лидер понял, что нужно почаще садиться и говорить с людьми: «Знаете, тут у вас не очень получилось. Давайте подумаем, как тут сделать лучше». Да, это не лучший вариант, здесь нельзя добиться волшебных результатов, но попробуйте – и все может получиться.

Эндрю: **И как это работает на практике? Допустим, я приглашаю вас, чтобы вы помогли мне улучшить управление моей командой. Сейчас наша команда – это я и *Дженни:* мы пишем книгу «Идеальные коман-**

ды». У нас есть команда примерно из 35 человек, специалистов по программированию, и все мы стремимся к одной цели — написать книгу, которая была бы одновременно информативной и интересной. Что бы вы сделали для нас в первую очередь?

Кеоки: Попросил бы вас рассказать мне побольше о себе и Дженни. Кажется, я о вас не очень много знаю. Итак, рассказывайте — прямо сейчас.

Эндрю: **Сейчас?**

Кеоки: Конечно. Что вы делаете? Какова ваша цель? Кто вы?

Эндрю: **Хорошо. Что ж, мы с Дженни начали сотрудничать с издательством O'Reilly несколько лет назад. Нам очень понравилось писать две книги для их серии «Head First»: одна — по проект-менеджменту, а другая — учебник по C#.**

Перед этим мы написали книгу о проект-менеджменте и разработке программ. Собственно говоря, это книга о качестве — об улучшении программ и программистских проектов. Мы считали в то время, что наша цель — сделать мир немного лучше, помогая людям лучше создавать программы. Вот это мы и делаем: мы помогаем людям делать программы лучше. Для этого мы пишем книги, выступаем на конференциях и часто консультируем.

В этом и причина того, что наши карьеры во многом связаны с управлением проектами. Мы довольно рано поняли, что менеджеры проектов — это недоиспользуемые и недооцененные ресурсы для создания хороших программ. Если у менеджера проекта хватает времени понять, как создается программное обеспечение, он может существенно помочь своей команде. Так что, пожалуй, это и есть наша цель: мы хотим предоставить людям инструмент, который нужен им для создания хороших программ и для того, чтобы помочь им решить проблемы, которые мешают им эти хорошие программы создавать.

В начале карьеры я долго программировал, а потом возглавлял команды, которые делали программы. Дженни сначала тестировала программное обеспечение, а затем возглавляла команды разработчиков, руководила ими. Я нечасто использую эпитет «блестящий», но она блестящий инженер по качеству. До встречи с ней я думал, что тестирование программ — это род черной магии, что-то врожденное или то, к чему ты чувствуешь склонность. Но когда я в 1990-е начал работать с Дженни, то понял, что тестирование программного обеспечения — это такой же род инженерной деятельности, как и все остальные. Им можно овладеть, его можно объяснить, можно набрать для него команду. И она делает это блестяще. Мне случалось отсылать ей,

как я считал, идеальные программы, но она находила в них столько ошибок, что я поверить не мог, что все это натворил я. Более того, работая вместе, мы поняли, как избежать большинства ошибок, поставив во главу угла программу. Это помогло мне постичь науку и искусство проверки качества, создания хороших программ.

Кеоки: У сотрудников отделов качества есть определенный образ мышления и типовая мотивация, нужно только понять это. Я сам начинал инженером по качеству в Microsoft, где и познакомился с отдельными представителями того типа, о котором вы, по всей вероятности, и говорите.

Эндрю: **Дженни была первым настоящим инженером по качеству, с которым я познакомился и который действительно глубоко понимал качество. Не на уровне «давайте сделаем так, чтобы эта программа не ломалась», а на уровне понимания того, как работает программа и насколько она удовлетворяет нуждам пользователей. Подлинное определение качества — это то, насколько программа соответствует требованиям. И это ее мотивация — да, собственно, и моя.**

Качество относится к процессу работы, но сам этот процесс — не просто принуждать людей делать свою работу определенным, из раза в раз повторяющимся образом. Это намного больше. Необходимо изменить то, как организация дышит и развивается, как работают люди, и — желательно — то, что они думают о своей работе.

Мы работали вместе несколько лет в небольшой компании, выпускающей финансовые программы, и методом проб и ошибок мы нашли такую компанию, которая была достаточно невелика и неопытна, чтобы дать двум относительно молодым и неопытным сотрудникам возможность изменить — полностью изменить — способ разработки программ.

Мы достигли довольно приличных результатов. По дороге у нас бывали и ошибки. И это пошло на пользу нам обоим. Мне кажется, я нахожусь в довольно выгодном положении благодаря тому, что сделал эти ошибки. И солидную часть нашей карьеры мы посвятили пониманию ошибок, пониманию того, почему проекты терпят неудачу.

Кеоки: У меня уже сложился в уме список ваших основных мотиваторов, и, возможно, главный из них – это отношение к ошибкам.

Есть талант, присущий почти каждому, при этом почти каждый считает его уникальным, чувствуя, что просто обязан поделиться им: это талант критика: «Я уверен, что тут что-то не так, поэтому я должен сказать, что тут плохо. Я чувствую себя обязанным». И тут складывается такая ситуация: никому это не нравится, но все мы в этом отношении просто асы.

Бывали ли вы в ситуации, когда, скажем, на встрече случилось что-то плохое? Неважно что. Допустим, мы сдали программу, а в ней оказалась критическая ошибка, и это ужасно.

Мы сидим и обсуждаем это, и кто-то говорит: «Надо проанализировать, что случилось, и выяснить причину». И тут звучат слова: «Это больше не повторится». Слыхали такое?

Эндрю: **Да, конечно. Как и большинство людей.**

Кеоки: Очень хорошо, а теперь вопрос. Возможен ли успех при отсутствии риска?

Эндрю: **Хороший вопрос. Полагаю, вряд ли.**

Кеоки: При нулевом риске нулевым будет и успех, поскольку если не рисковать, то и компенсации не получишь. Без риска нет успеха. Но что такое риск? Риск – это возможность того, что случится что-то плохое. Согласны?

Эндрю: **Абсолютно.**

Кеоки: Так вот, если успех не бывает без риска, а риск означает возможность того, что случится что-то плохое, то если вы создаете среду, в которой риск сводится к нулю, то что еще сводится к нулю? Возможность успеха.

Эндрю: **Никогда ни о чем подобном не думал, но звучит логично.**

Кеоки: Так что неплохо бывает сказать: «Знаете, мы, возможно, неправильно оценили возможность риска и что-то пошло не так. Посмотрим, сможем ли мы здесь что-то исправить, чтобы в будущем быть более информированными».

Но затем, когда ошибка произошла, вы ищете ее виновника. На глазах у всех его голова катится по коридору и вылетает из ворот компании. И все понимают: «Нельзя допускать ошибок». Тем самым вы порождаете ситуацию, когда все боятся допустить ошибку и никто не хочет идти на риск.

Вместо этого после совершения ошибки надо создать другую эмоциональную среду. Вы говорите: «Так, что за ошибка? Хорошо, мы ее допустили. Отличная ошибка! Больше мы так делать не будем. Теперь будем обращать на это больше внимания». Можно даже выписать ошибившемуся премию.

И что потом? Это порождает атмосферу, в которой люди не боятся идти на риск. Вы хотите воодушевить сотрудников. И теперь, идя на риск – а разумный человек, владея всеми фактами, способен оценить риско-

ванность решения, – они не боятся, что сразу вылетят с работы. И никто не считает это странным.

Если кто-то принял разумное с виду решение, чреватое плохими последствиями, это означает, что он рисковал, то есть знал, что может случиться что-то плохое. И знаете что? Иногда плохое и случается. Но если вы от этого не беситесь, а принимаете как данность, то ваши подчиненные видят, что могут продолжать искать новые возможности. А новые возможности приводят к инновациям, так и рождаются великие проекты.

Вот почему часто успешные компании начинают с больших возможностей. Они добиваются успеха и всю оставшуюся жизнь пытаются свести возможность успеха к нулю, опасаясь потерять то, что имеют. А сами попытки не потерять обречены на провал.

7

Музыкальную индустрию – в XXI век!

Одна тяжба за раз

Том Тарка

Мы приземлились в лос-анджелесском аэропорту, и через две минуты по громкой связи сообщили: пассажира Томаса Тарка просят подойти к сотруднику аэропорта.

Я навострил уши, и несколько нетвердый разум воспрял духом, начав проигрывать все возможные сценарии.

Кто-то из моих дедушек и бабушек смертельно болен, и надо срочно позвонить родителям?

Контрабандисты подсунули в мою сумку наркотики, а собаки из Международного терминала их уже вынюхали?

Я отъехал на автобусе вместе со всеми, уже выведенный из себя простой фразой по громкой связи и не способный ни на что, кроме как тащить свою переполненную сумку и ждать целую вечность, как всегда кажется после долгого международного перелета.

Вот так я вернулся в США, обогнув весь земной шар...

За десять месяцев до этого я жил в Боулдере и стучал по клавишам в различных качествах: сначала был тестером программного обеспечения, потом продвинулся по пищевой цепочке до веб-мастера небольшой программистской компании. Я не был сильно привязан к Боулдеру, и совет туристов из гавайского хостела насчет относительно недорогих авиабилетов вроде «Вокруг света» побудил меня сэкономить денег, стереть пыль с паспорта и посмотреть, как живут в остальном мире.

Что касается работы, то я готов был ее сменить, не в последнюю очередь потому, что большую часть своей карьеры я работал в основном один, будучи единственным представителем техподдержки в проекте. Я чувствовал себя одиноким, а мне хотелось общения. Например, работать веб-мастером в той компании было интересно, но сайт принадлежал отделу маркетинга, и мне приходилось работать с хорошей, но устаревшей платформой, которая не имела отношения ни к UNIX, ни к программированию. Сказалось, наверное, и то, что многие мои друзья переехали в район Залива и часто пели песни сирен о доткомах, о толпах гиков, которые остаются программировать на всю ночь, создавая «Клевые штуки™» и порой даже на этом обогащаясь. Я был готов переехать, и я был уверен, что теперь мне следует поступить программистом в какой-нибудь дотком. К тому же был экономический бум, люди с моим набором навыков требовались везде, и я воспользовался тем, что можно уехать на несколько месяцев и, судя по всему, без проблем устроиться на новую работу. Итак, в сентябре 1998 года я ушел с должности веб-мастера, собрал пожитки и двинул через всю страну в Нью-Йорк, вскочив на самолет в Лондон через 48 часов после отъезда из Колорадо.

Спустя 10 месяцев и 45 минут я сидел на заднем сиденье универсала посреди Лос-Анджелеса, два незнакомца везли меня куда-то, а я с удивлением поглядывал вокруг: восьмиполосные шоссе, пробки, христианские плакаты, уверявшие меня, что я никак не могу выбраться из пробки, потому что произношу имя Господа всуе; забавные американские машины всех форм, отполированные и блестящие, неуклюжие здания в никак не заканчивающейся застройке... Лос-Анджелес – нет, сама *Америка* нахлынула на меня впервые за десять месяцев. Она была больше всего, что я видел за всю жизнь, – нет, не больше, но просто *иной* – каким-то неописуемым образом, который отражал такой образ жизни, который был для меня во многом чуждым. Я был дома, но мой дом уже не казался мне родным.

Водитель машины был приятелем Конни-Линн, которую я знал заочно уже года два, но никогда не встречал лично. Именно она позвонила в аэропорт, чтобы мне передали такое послание: «Сообщение для Томми Тарка. Констанс не сможет забрать тебя из аэропорта, но пошлет кого-нибудь, кто встретит тебя у ворот». Итак, машина ехала в Пасадену, вел ее человек, которого я не знал, а вез он меня к человеку, которого я никогда не встречал лично и у которого я собирался остановиться на пару дней, пока из Финикса не приедет Марк.

Марк, разумеется, был мой старый приятель по колледжу и путешественник, он ехал через всю страну из Пенсильвании и должен был встретиться со мной в Лос-Анджелесе, проведя перед этим некоторое время с семьей в Финиксе. Он только что окончил магистратуру в Пен-

сильванском государственном университете, и мы намеревались неспешно ехать по стране, изучая Соединенные Штаты, пока не доедем до Восточного побережья. В это время мои пожитки (в том числе машина) прозябали на чердаках, в сараях и гостиных трех городов.

Конни-Линн оказалась такой же замечательной и милой, какой я и ожидал ее увидеть после двух лет онлайн-знакомства, мы встретились с Марком и через три дня уже были у друзей в Сан-Диего: Джей Ви и Ка Эл Эйч были нашими знакомыми из разных времен и мест, но как-то независимо оказались там. Мы побросали фрисби, а потом вернулись в комнату Ка Эл Эйч в кампусе Университета Сан-Диего, и тогда я сказал:

«Знаете, я бы не возражал задержаться в Сан-Диего на пару месяцев, тем более что свадьба Джей Ви уже через несколько недель, а если я уеду, то почти наверняка на нее не вернусь. Надо бы найти какую-то работу, так что дайте знать, если найдется временное местечко сисадмина или спеца по UNIX».

После путешествий, переезда с места на место по расписанию, которое определяется погодными условиями или временем отбытия следующего автобуса, желание найти что-то более стабильное кажется совершенно естественным. И я произнес эти слова с полной ответственностью: я собирался зависнуть в Сан-Диего на шесть недель до свадьбы приятеля, но никак не ожидая, что из этого выйдет что-то более серьезное. Кроме того, после 10 месяцев путешествий у меня почти кончились деньги, так что у меня не было возможности жить в Сан-Диего без работы, и желательно прибыльной, чтобы набить карманы. Итак, все упиралось в финансы.

Вот почему я удивился, когда услышал ответ: «Да? А ты говорил это Джей Ви? Наш друг Дэвид работает в стартапе, и им, думаю, нужны люди. Что скажешь?»

О господи!

Так и порешили. Мы с Марком вернулись в Лос-Анджелес, но через три дня я сел на Surfliner – поезд, который по Тихоокеанскому побережью идет в Сан-Диего, – а потом поймал такси на интервью в MP3.com, компанию, о которой я не знал абсолютно ничего. Абсурдность ситуации – мой первый свежий взгляд на Америку и способ приема на работу типа «спроси вселенную – и воздастся тебе» – усугублялась тем, что накануне я забыл сумку с единственным набором приличной одежды в багажнике чьей-то машины. Я прибыл на интервью в лучшем, что у меня было с собой: в шортах, несколько помятой, но на двух пуговицах рубашке с воротничком и узким вырезом, украшенной рекламой пива и главой из устава клуба «Hash House Harriers», и, насколько помню, в сандалиях. Собственный вид меня несколько угнетал, но ранний час поездки

(то есть все было закрыто), невозможность легко пересесть (ехал-то я на общественном транспорте) и недостаток времени между пересадками сделали свое дело: все, что я мог, – это войти, сделав хорошую мину.

«Да, мы принимали ставки, сколько ты выдержишь», – сказал Джон Бек, технический директор, как выяснилось позже на собеседовании.

В тот день MP3.com кипела. Назавтра было назначено первичное размещение акций, и все носились как сумасшедшие с приклеенными к ушам сотовыми, сражаясь с недавней аварией неизвестной природы, которая произошла из-за внезапной атаки, вызванной шумихой в прессе и последовавшим веб-трафиком. По меньшей мере половина моих собеседований прошла на скамейке в парке в условиях всегда прекрасной погоды Сан-Диего: инженеры, которым, судя по всему, и так было чем заняться, глубоко вдыхали, прежде чем пытаться спокойно разузнать что-то об этом сумасшедшем парне только что с самолета («Откуда? С Фиджи?»), а технический директор, выглядевший на тот возраст, когда еще официально нельзя пить, задавал мне вопросы по модулям Perl (я знал ответы на его вопросы по DBI, но в то время еще не был знаком с Carp).

Если вы думаете, что за 10 месяцев бродяжничества мои навыки ослабли, то ошибаетесь. Я довольно много времени провел, развивая их: сидел в Сингапуре в библиотеке, читая книги по искусственному интеллекту, временно работал программистом или сисадмином, чтобы пополнить кошелек (или просто чтобы получить бесплатный кофе и выход в Интернет, как в том интернет-кафе, бывшем по совместительству офисом программистов с открытым исходным кодом: большинство его сотрудников служили разработчиками и где-то еще), познавал достоинства и недостатки DBI, поддерживая собственный веб-сайт. Я вернулся с пробелом в трудовом опыте, но программистские навыки существенно улучшились, притом именно те, как выяснилось, которые были наиболее необходимы MP3.com. Иногда мне кажется, что те дни, которые я провел в Бонди на балконе, наслаждаясь солнцем и разбираясь в коде Perl на компьютере 386/16 (12 или 16 мегабайт RAM, модем 33K), дали мне больше, чем если бы я резвился на пляже. Как-то так.

Через четыре часа на столе уже был контракт, и мы обсуждали зарплату и премии. То, что первичное размещение акций было лишь на следующий день – а настоящий договор до того времени не мог быть подписан, не говоря уж о том, что мне нужна была по меньшей мере неделя, чтобы перевезти барахло с Восточного побережья, – означало, что мне могли предложить лишь фондовый опцион: вместо того чтобы получать доллар-другой со сделки, я согласился на долю с рыночной цены (мой фондовый опцион был примерно 50–65 долларов): можно было быстро обогатиться. Я был несколько напуган, но в то же время понимал, что это мой козырь по сравнению с обычной зарплатой – гро-

шами, на которые кое-кто соглашался, – полагаясь, разумеется, на ее незыблемость. И я сделал все возможное, чтобы воспользоваться сложившейся ситуацией к своей выгоде. Я вышел с меньшей зарплатой, чем хотел, но с обещанием, что через несколько месяцев мне предложат что-то более достойное. Вполне нормально, учитывая, как сильно мне была нужна работа.

Итак, меньше чем через две недели после возвращения в страну я умудрился устроиться на такую работу, о которой мечтал: работу в быстро развивающемся, расширяющемся доткоме. Немного жаль было, что я так и не поездил по Штатам, да еще и оставил Марка в одиночестве, учитывая наши планы ленивого исследования, но это была возможность всей жизни. Шанс на поездку еще мог и представиться (и представился); но теперь у меня была работа!

Новый проект, новая команда

Быстро пролетели три месяца, и MP3.com переехала в два здания с видом на живописное шоссе 805. Только в одном из двух зданий были люди, да и оно еще отделывалось: офисы находились на пустых пространствах, здание окружал забор, чтобы сотрудники никого не укусили, и оно было украшено: один проход был расписан граффити, а второй текстурирован тяжелым уплотнителем и выкрашен в золотистый цвет, похожий на цвет печенья с арахисом. Представьте только: эти здания теперь наши. Второе из них еще строилось и вскоре должно было принять инженерный отдел, в котором уже было 50–60 человек – около 40–50% всего численного состава компании. Пока, однако, приходилось ютиться в чистилище.

Начало октября и три предыдущих месяца я в основном работал с Сандером Ван Зюстом, сумасшедшим голландцем, одним из основателей MP3.com, известным как Великий хакер, и Дэвидом Стори, выдающимся программистом. У них столько же сходных черт, сколько и различных. Сандеру уже 23 года, у него взъерошенные синие волосы, он, кажется, никогда не спит, и никто не удивится, встретив его на техновечеринке в Сан-Диего, куда он наверняка явится в серебристых трусах. Он поражает как своей манерой говорить – всегда ждет, когда вы закончите, убеждается, что вам больше нечего сказать, а потом осторожно дает вам проницательный совет по поводу того, почему система сделана именно так, как данная архитектура прошла цикл разработки и получилась именно такой, что имеет смысл добавить в следующую версию, – так и своим слегка не американским поведением (но в определенном смысле – все-таки американским).

Дэвиду, напротив, всего 39 лет, и блеск в его глазах выдает, что он бы с большим удовольствием весь день разъезжал с детьми на велосипедах BMX, чем жил во взрослом мире. Но он не только никогда не устает и рассматривает с любопытством каждый новый проект, радуясь каждый раз, когда отгадывает загадку, но и излучает, казалось бы, непробиваемое спокойствие. Он спортивный, одевается в элегантные, выглаженные рубашки с воротничками, а на работу в основном ездит на своем мотоцикле для кроссов, едва ли не запрещенном для езды по городу.

Их поведение, внешний вид и трудовые привычки соотносятся как инь и ян, но оба они – законченные профессионалы: создают элегантные конфигурации, пользуются имеющимися стандартами, если таковые есть (или, по крайней мере, сознают, что уклоняются от действующих стандартов и почему), пишут и подписывают комментарии, когда в них есть смысл, и обычно не торопятся сдать конфигурацию, если ее можно легко улучшить позже.

Кажется, это одна из общих черт всех инженеров MP3.com: все они точь-в-точь талантливые профессионалы, не боящиеся засучить рукава и выступить с новаторскими решениями. Не передать, сколько новых сотрудников MP3.com первый день или даже неделю ничего не понимали, потому что вокруг все двигалось слишком быстро. Однако затем происходило нечто поразительное и даже уникальное: они брались за работу сами, для себя. Они спрашивали не «Что мне нужно сделать?», а «Что я могу сделать??!», поскольку хотели быть частью живого функционирующего вокруг организма. Так что главным образом люди приходили в MP3.com именно работать. Они были в восторге от того, что делает компания, от собравшейся команды, от сумасшедшего темпа во всем: такая энергия вызывала даже привыкание. Вот так я и стал работать с Дэвидом и Сандером.

> «Как ты думаешь, где будешь слушать свою музыку через 10 лет?» – Майкл Робертсон, исполнительный директор MP3.com.
>
> «В тюрьме», – Кристофер Джайлс, инженер-программист и скрипач.

Итак, в начале октября мы втроем с Дэвидом и Сандером пришли в конференц-зал, где уже были наш менеджер Дэвид Макколлум и три других инженеров: Кристофер Джайлс, Джош Стивенс и Майк Олифант.

> «Послушайте, это сверхсекретный проект, и то, о чем мы будем говорить, не должно пересечь порог этой комнаты. Хорошо? Вы знаете, что мы некоторое время вели переговоры с музыкальными лейблами, чтобы выработать решение по лицензированию их продукции. К сожалению, переговоры зашли в тупик. Только что я был у Майкла [Робертсона],

и у него есть идея, как можно легально продолжить свою деятельность. Он хочет создать сайт, который позволял бы людям обмениваться на нем их музыкальными коллекциями, то есть теми дисками, которые у них уже есть...» – Дэвид Макколлум, главный инженер.

Что-о-о??!

«Насколько это законно?» – спросил кто-то.

«Майкл и юристы изучают вопрос и считают, что все тут легально...»

И началось.

Видите ли, до того времени MP3.com был веб-сайтом для независимых исполнителей. Любой музыкант мог создать свою страничку, закачать музыку – и вперед! Музыку можно было слушать и (если музыканты не против) скачивать. Все бесплатно! Кроме того, можно было закачать альбом и продавать на нашем сайте диски, отдавая 40% цены. Эти DAM-диски печатались с ходу, не стоили вам ничего, а наши расходы покрывались процентом от продаж.

В общих чертах все так и было: сайт был создан для самопродвижения музыкантов, снижения стоимости создания музыкального продукта и устранения посредника (то есть звукозаписывающего лейбла), который получал огромные прибыли, – но здесь не отражены возбуждение и шум, сопровождавшие работу сайта. Не отражено тут и то, что было повседневной рутиной компании: весь штат музыковедов в дополнение ко всем остальным обязанностям без устали прослушивал трек за треком закачанной музыки на предмет нарушения авторских прав (и поверьте, эта музыка не всегда была приятна!), отделы продаж и маркетинга каким-то образом пытались снизить наш бурно растущий рейтинг, юридический отдел делал все, что мог, чтобы уберечь нас от проблем (мягко говоря, непростая задача), ну и, наконец, инженерный отдел ежедневно поддерживал работу сайта.

Говоря «инженерный», я имею в виду инженеров. Поддержка сайта, на котором размещены 200 тысяч песен и 40 тысяч исполнителей, – нелегкая задача, и MP3.com, обладая 400 миллионами долларов в банке, вкладывалась в специалистов по качеству и инженеров, которые составляли половину штата компании.

Что все это значило? Что ж, через три месяца после открытия инженерный отдел – все его сотрудники – сделал существенную заявку на создание мощной инфраструктуры. Поймите меня правильно: лихорадочная поступь и энергия не уменьшились, но все стало... стабильным. И заметьте, что я говорю здесь «инженерный отдел», не упоминая отдельных личностей, как до сих пор. Звучит странно, но, даже несмотря на довольно большое число инженеров (50 или 60), коллектив оста-

вался очень сплоченным, по крайней мере все результаты трудов одной группы проходили проверку со стороны других.

Собственно, и вся компания была сплоченной: еженедельно устраивались встречи для всех сотрудников, где наш исполнительный директор Майкл Робретсон выступал перед всеми, представлял новичков и знакомил нас в общих чертах с «масштабной перспективой». Даже если вы не знали всех сотрудников компании, не знали всего, что происходит, вы все время понимали, что делает руководство, – по крайней мере, понимали, что оно *что-то* делает. Можно было *не соглашаться* с этими действиями, но тем не менее было понятно, что они чем-то заняты. Поймите меня правильно: эти «беседы» порой были накачками, порой пропагандой, чтобы люди сосредоточились на работе, но такая стратегия была действенной. Как в семье, мы сплачивались друг около друга, особенно когда ситуация начала меняться к худшему. Но до этого было еще далеко...

И еще слушать Майкла было всегда интересно, потому что он был провидцем. С его предвидением или реализацией идей можно было не соглашаться, но он мог мыслить масштабно и видеть вещи в перспективе. Порой это могло выйти боком, потому что он обычно фанат своей идеи и не стесняется указать, что вы неправы. Поэтому общаться с ним бывает страшновато, поскольку всегда непонятно, чего от него ждать: может быть, ваши идеи придутся Боссу по сердцу и вам дадут карт-бланш? Или Майкл начнет кричать и ругаться так, что с ним нельзя будет спорить, – если вы, конечно, цените свою работу, – и останется глух ко всем доводам разума, как бы тщательно вы ни подбирали слова и каких бы свидетельств в свою пользу ни приводили: его слово всегда будет решающим.

Но именно это и делало Майкла по-своему великим: его страсть, его ясное видение ситуации, способность представить продукт во всей полноте и выразить это представление, взять крупицу идеи и исследовать ее, выдавать одну идею за другой, пока она не превратится в продукт. Неотъемлемы были его уверенность и спокойствие, позволяющие проводить эти идеи в жизнь. Эмоционально это могли быть настоящие американские горки, но нет большего эмоционального подъема, чем когда ваши идеи претворяются в жизнь.

Итак, когда Дэвид Макколлум стал рассказывать нам шестерым об идее Майкла, в каком-то смысле это не стало громом среди ясного неба. Майкл понимал, в каком направлении должна двигаться компания, как увеличивать свои доходы, и Дэвид просил нас оценить эту идею и решить, сможем ли мы ее воплотить. Однако сама идея была слишком свежей и расплывчатой. Суть ее сводилась к «сервису на трех опорах: 1) вы можете пользоваться уже имеющейся у вас музыкальной

коллекцией (ничего заново приобретать не надо); 2) если вы купили новую музыку, ее можно тут же прослушать; 3) вы сможете слушать вашу музыку, где бы вы ни находились, на любом устройстве, через которое вы хотите ее воспроизводить».

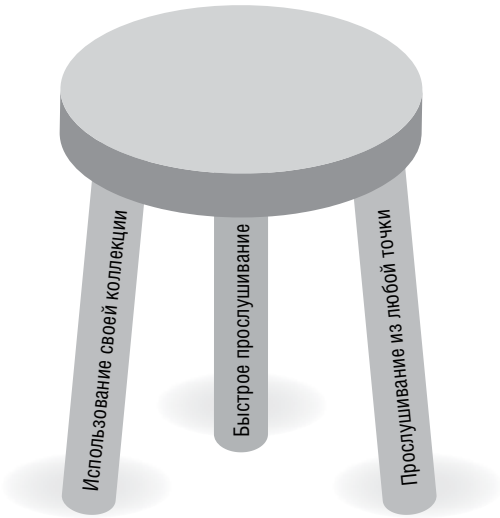


Рис. 7.1. *Онлайн-сервис MP3.com позволяет вам собрать всю свою музыку в одном месте, доступном из любой точки мира*

Мы внимательно слушали, пытаясь понять, к чему это приведет, и Дэвид немедленно пояснил: мы должны будем конвертировать (то есть скопировать музыкальный файл с компакт-диска на жесткий диск пользователя) огромный массив компакт-дисков, и если пользователи докажут, что у них есть этот CD, мы предоставим им доступ к нему на нашем сайте.

Чтобы объяснить ряд мест, которые кажутся сомнительными, – например как вяжется конвертирование десятков тысяч компакт-дисков с любой законной бизнес-моделью, в то время как во всей стране (а может, и во всем мире) Звукозаписывающая ассоциация Америки (RIAA) с пеной у рта отстаивает то положение, что даже создание MP3-файла для любых целей является незаконным, нам придется вернуться в старый добрый 1976-й год, когда Sony выпустила первый пишущий видеомагнитофон Betamax. Говоря вкратце, на Sony немедленно подали в суд за нарушение копирайта, поскольку Betamax мог записывать телешоу, фильмы и все, что идет на телевидении. Ранее такое было невозможно, а теперь рассматривалось как нарушение прав телекомпании. Sony выиграла суд, и после законодательного решения появилось понятие «вре-

менная манипуляция», которое означает, что пользователь может сохранить лицензионный контент на неопределенный срок, чтобы использовать позже. Эта зацепка известна также как «Решение по Betamax».

Следствием из временной манипуляции стала пространственная манипуляция, то есть возможность сделать копию того, что у вас уже есть – CD, DVD, записи, – на другой носитель, например с CD на кассету. Эту идею в деле против производителей MP3-плееров – Diamond Multimedia Systems – безуспешно оспаривала RIAA, пытаясь доказать, что копирование музыки с CD на MP3-плеер – это нарушение авторского права.

Рис. 7.2. *Плакат MP3.com – часть широкой пропагандистской кампании RIAA по борьбе с музыкальным пиратством*

На основе этих прецедентов кто-то (вероятно, наш юридический отдел) одобрил возможность того, что мы можем конвертировать ряд компакт-дисков и дать к ним доступ пользователям, которые могут доказать, что уже владеют этим контентом. Мы просто позволяли им осуществить «пространственную манипуляцию» уже имеющимися у них записями. После кратких споров Джог Стивенс заявил: «Мы все прикидываемся юристами и пытаемся разработать такую систему, которая будет безопасной и не даст доступа к музыке тем, кто не является ее обладателем». Понимаете, с нашей точки зрения мы считали, что, пока система останется безопасной, на нас не подадут в суд, но если система будет с изъяном и они (RIAA) решат, что к музыке имеется незаконный доступ, то у нас будут проблемы.

Нет, по поводу возможных исков все равно ворчали, даже мы шестеро. Возможно, я был одним из немногих, кто твердо верил в «линию партии» и легко дал себя убедить, что то, во что мы ввязываемся, законно.

Нам сказали, что юристы все проверили, а раз я не юрист, то и ладно. Вот так-то! Возможно, впрочем, тут были замешаны и другие факторы, например атмосфера нереальности, окружавшая многие наши проекты в MP3.com. Я главным образом просто был чертовски счастлив работать в таком окружении, с таким количеством талантливых ребят.

ПЛАН

1. Конвертировать 10 000 самых популярных компакт-дисков (на основе истории продаж).
2. Создать веб-интерфейс для управления контентом.
3. Создать программу, позволяющую пользователям доказать, что компакт-диски их собственные (Beam It).
4. Найти розничных партнеров для обеспечения «моментального прослушивания».
5. Разработать систему выплаты роялти непосредственно музыкантам.

И мы начали работать над системой, которая позволяла бы пользователям управлять своей музыкой вне зависимости от того, были ли это композиции с MP3.com или с крупных лейблов (диски которых нужно было предварительно купить). Система была основана на существующем сайте My.MP3.com, который позволял нечто подобное, но только по отношению к музыкантам с MP3.com. My.MP3.com должен был подвергнуться косметическому ремонту, и нужно было значительно добавить функциональности, чтобы закрыть новый контент, но как плацдарм он был вполне неплох. Кроме того, мы вторгались в новую сферу программирования, создавая инструмент, который могли бы скачать пользователи My.MP3.com, чтобы впоследствии идентифицировать себя как владельца данного диска.

Новый веб-сайт по-прежнему должен был называться My MP3.com, но кодовое название проекта было Da Bomb: отсылка как к атомной бомбе, поскольку мы ожидали, что этот сервис навсегда изменит музыкальную индустрию (как военную промышленность навсегда изменила атомная бомба), так и к песне P-Funk – привязка к музыке (ну и показатель того, что мы крутые, блин!)

Независимо от этого, но примерно в то же время, когда велась работа над Da Bomb, я начал участвовать еще в одном проекте вместе с другим инженером, Брайаном Дегенхардтом. Это касалось отслеживания пользователей и параллелизма. Если вкратце, то у MP3.com не было

возможности узнать, проигрывает пользователь песню в данный момент или нет. Мы разработали систему броузерных строк данных, так что знали идентификационные номера пользователей в системе MP3.com, но сервер не мог определить, вошли ли вы в систему на нескольких компьютерах одновременно или нет; он просто фиксировал, что вы в определенное время обращались к определенной песне. Этим проектом нас никто не просил заниматься – собственно, многие говорили мне, что нам это и не нужно, – но мы решили, что все равно это сделаем. Я опросил тех, кто, как мне казалось, мог нуждаться в этих данных, и потом мы с Брайаном поработали несколько вечеров допоздна у доски, определяя требования к проекту и пытаясь понять, как будут работать вместе отдельные элементы. Это было круто: мы оба были довольно молоды, и это был, несомненно, только первый наш «большой» проект, который мы начали создавать с листа. Вокруг было много шуму: мы изменяли то, как вся музыка с нашего сайта будет попадать к конечным пользователям. Сами. Потому что нам *разрешили*. Это было круто.

С Брайаном было замечательно работать. Он был абсолютным новичком, прямо из колледжа, и у него был эдакий позитивный взгляд на мир, который выражался у него в том, что он с энтузиазмом мог распространяться о самых продвинутых свойствах Apache перед всеми, кто готов был его слушать; он мог выступать с самыми забавными, но притом чертовски точными аналогиями, которые сводили происходящее к глупейшим карикатурам: «Смотри, вот у нас слон, и он хочет подобрать вот этот орех, но не может, потому что...» Да-да. Потоки слов не останавливались, пока он не разбирался, почему Apache работает не совсем так, как он от него хочет. Честно говоря, понятия не имею, почему мы взялись за этот проект вместе. Сандер, вероятно, был по уши в других делах, связанных с его титулом Великого хакера, и либо это он направил меня к Брайану, либо мы с Брайаном сами как-то стали переговариваться через стенку, потому что в основном засиживались на работе допоздна. Такая уж была работа в MP3.com: возможность работать над самыми безумными проектами с самыми крутыми людьми, понять, как проблема решается, а затем *решить* ее, разобравшись с подробностями по дороге.

Рассчитанный риск

Декабрь, и работа над Da Bomb в полном разгаре. Тайна уже вышла наружу – по меньшей мере, внутри компании: несколькими неделями ранее технический вице-президент компании Делон Дотсон объявил о проекте всему инженерному отделу, а Майкл вскоре после этого сообщил о нем на общем собрании коллектива. Теперь MP3.com превратилась в сборище меломанов-маньяков, покупая все доступные компакт-

диски: сотрудникам предлагали 20 долларов за право использовать CD из их личных коллекций. Целью было приобрести 10 000 самых популярных компакт-дисков в истории, так что когда пользователи сайта покупали у поставщика-партнера компакт-диск или доказывали, что уже имеют такой диск в коллекции, был большой шанс, что у нас в базе данных эта музыка уже есть. Это оказалось более серьезной проблемой, чем мы ожидали, так как выяснилось, что у каждого CD может быть несколько версий. Наша программа для верификации Beam It оказалась к этому чувствительной, и когда вы пытались верифицировать компакт-диск из вашей коллекции, а у нас была не та версия альбома, вы не могли ее прослушать с наших серверов. Это было хорошо с позиции безопасности – нельзя было слушать музыку, которой у вас нет, но плохо с точки зрения конечного пользователя: «Не могу поверить, что у них нет „Dark Side Of The Moon“!» А меж тем есть как минимум восемь версий этого альбома, и даже если на слух некоторые из них кажутся идентичными, то мастеринговые пленки слегка отличаются: этого достаточно, чтобы данные при сравнении файлов не совпали.

Оказалось, что 10 000 компакт-дисков – это не просто симпатичное круглое число: мы не знали, что Робин Ричардс, президент MP3.com, одобрил Da Bomb только на том условии, что 10 000 дисков будут изначально загружены в систему. Почему именно 10 тысяч? Просто Робин произвел подсчеты, и выяснилось, что, имея 400 миллионов в банке, мы даже при условии штрафа от 10 до 25 тысяч долларов за нарушение авторских прав в каждой песне, почему-либо проиграв процесс, все же сохраним в банке некоторую сумму. Это рассчитанный риск, но на него стоило идти: если бы проект с My.MP3.com удался, то MP3.com захватила бы уверенное лидерство во всей индустрии.

Тем временем команды разработчиков и специалистов по качеству на полном ходу продолжали работать, и начинала сказываться усталость. Кристофер Джайлс был единственным инженером, разрабатывавшим новый сайт My.MP3.com – поскольку он занимался предыдущей версией сайта, – и стоявшая перед ним задача подавляла: нужно было полностью переделать пользовательский интерфейс, добавить запись и управление действиями пользователя, неоднократно оптимизировать сценарии, чтобы они не подвешивали базу данных, не грузились дольше пары секунд и так далее. Помимо всего прочего, было очевидно, что Кристофера – который и так-то не был образцом спокойствия – сильно раздражал напористый, ориентированный на детали (это эвфемизм для слова «въедливый», что ли? Не знаю) стиль руководства Дэвида. В свою очередь, Дэвид должен был контролировать, чтобы все шло по расписанию, и он не стеснялся идти на конфликт, если этому расписанию что-то угрожало.

Все дошло до ручки, когда Кристофер обозвал Дэвида «диктатором», и через минуту они исчезли в конференц-зале. Находясь в пяти метрах от него, я слышал разгоряченную дискуссию из-за закрытых дверей. Через десять минут они появились, и стало ясно, что Кристофер недоволен: не то чтобы это было чем-то необычным, потому что он вообще был хмурым (хотя вполне приятным парнем, если узнать его получше и/или не иметь необходимости с ним работать), но в этот раз он был еще более недоволен, чем обычно. Дэвид не казался сильно выбитым из колеи, но по нему вообще мало что поймешь. (Дегенхардт говорил про него: «Дэвид Макколлум – это загадка, обернутая в тайну», – и не думаю, что он сильно ошибался.) И если честно, то за два года работы с Дэвидом это, думаю, был единственный раз, когда он вышел из себя. Не то чтобы он не мог гнуть свою линию или настаивать на том, что он хотел (или на том, что вы неправы), но у него была поразительная способность избегать прямых конфликтов с людьми: обычно он уводил дискуссию на такой уровень абстракции, что всем приходилось с ним согласиться. И мне стало любопытно: что же такого сделал Кристофер, что даже Дэвид потерял контроль над собой? Был ли это какой-то прямой вызов ему и его авторитету? Или же что-то еще более серьезное, потому что я никогда не видел, чтобы Дэвид травил кого-то так же, как Кристофера.

Надо отметить одну вещь, о которой мне говорили многие, в том числе Дэвид, и она относится к процессу найма в MP3.com. В то время MP3.com набирала пять-шесть инженеров в неделю, и основными вопросами после того как кандидат проходил несколько кругов собеседования, были: «Можете ли вы работать с этим человеком?» и «Уважаете ли вы их интеллект?» Хотя сами по себе эти вопросы вряд ли сильно отличаются от того, что можно услышать в других местах, видимо, в MP3.com они значили несколько больше обычного. Они проходили красной нитью через все собеседования, и в результате многие замечательные кандидаты были отсеяны, потому что оказались «болтами-восьмигранниками», которые не подходили MP3.com.[1] Требовалось соответствие многим критериям, из которых главными были интерес к проектам с открытым исходным кодом и/или желание научиться работать в них, но руководствовались также и личной совместимостью, и желанием быть не ниже высшего уровня.

Во многих отношениях Кристофер не соответствовал корпоративной культуре MP3.com и казался скорее одиночкой и чужаком. Поэтому я предполагаю, что, возможно, конфликт между ним и Дэвидом мог быть вызван недостатком общего уважения, происходившим из различ-

[1] «Болт-восьмигранник» – настолько прекрасный термин, что я не могу его присвоить: придумал его Дэвид Дудас.

ных взглядов на мир. Этот недостаток уважения шел с обеих сторон: подозреваю, тут-то и крылся, по крайней мере отчасти, источник напряжения. Не стоит упоминать, что этот эпизод и стал для Кристофера началом конца: через несколько месяцев ему пришлось уйти в Microsoft.

Приготовьте ваши конверторы, джентльмены!

Конвертирование 10 000 компакт-дисков – это геркулесов труд сам по себе, но прибавьте к нему необходимость точно вносить все данные всех альбомов в общую базу – и задача станет просто смешной. Каждому сотруднику компании предстояло обработать примерно 50 дисков, если этим будут заниматься все 200 человек, но это если каждый диск конвертировать только раз и больше не проверять данные, чего как раз не хотелось, так что в итоге вышло по 100 дисков на человека при условии общего участия. Все и участвовали. Майкл как-то в пятницу на общем собрании объявил, что нам нужна помощь всех, и надо же, думаю, что все сотрудники действительно стали над этим работать: по очереди приходили в импровизированную лабораторию по конвертированию и использовали это задание как отдых от повседневной рутины, а потом забирали диски на свое рабочее место, чтобы поработать над ними позже. Даже Майкл принял участие, а Робин Ричардс – нет: его не было в городе.

Где-то недели за две мы не спеша продрались сквозь все компакт-диски, которые компания приобрела в магазинах и у сотрудников. Их было устрашающее количество, и они обеспечили отличный тимбилдинг: офисный администратор трудился рядом с вице-президентом по продажам, скромная массажистка Линда – рядом с Парадайзом, одним из наших ведущих музыковедов и по совместительству отцом-основателем хип-хопа. Все очень волновались, и мы, хотя и по-прежнему беспокоясь насчет законности наших действий, стали обретать ту сплоченность, которую потеряли было, увеличив штат с 60–70 человек полугодом ранее до почти 200.

Только я не уверен, что в то время все понимали, что мы конвертируем не 10 000 дисков.

Когда вернулся Робин, он был в шоке: на нескольких файловых серверах лежало уже более 40 000 конвертированных дисков, которые ждали только доступа к ним пользователей с собственными CD, а этот доступ открывался сразу после открытия сервиса My.MP3.com. Я не помню, когда и как это было обнаружено: то ли Робин вернулся и увидел сотрудников за работой, но решил, что это те 10 000 дисков, о которых говорил он, и только потом обнаружил, что конвертировано уже 40 000? Понимали ли инженеры, что имеется жесткий лимит в 10 000 или считали,

что 10 000 – это только цель, которую можно превзойти на волне трудового энтузиазма? Или Майкл велел просто конвертировать все, прекрасно сознавая, что дисков гораздо больше, чем 10 000?

Уверен, что кто-то об этом знает, но в то время особой разницы уже не было: когда My.MP3.com был запущен, было конвертировано около 40 тысяч дисков – в четыре раза больше указанного количества, в четыре раза потенциально лучше, но достаточно, чтобы выбить нас из бизнеса! Если мы не хотели удалять все эти альбомы – а их действительно пришлось бы удалять и потом делать все заново, теряя недели работы, – то ставки игры становились на порядок выше. Но мы не обращали внимания. Мы собирались изменить мир и воевать со злом – музыкальными корпорациями. В Da Bomb заключалось будущее музыки, а основные лейблы просто не хотели сдавать позиции. Нам предстояло перенести музыку в XXI век, и мы собирались сделать это законно: просто дать пользователям возможность пространственного перемещения того контента, который они уже приобрели. И что мы сделали? Продолжили конвертировать.

Последний месяц

Тем временем работой над Da Bomb уже были заняты все сотрудники инженерного отдела. Джеймс Парк был главным по созданию пользовательского интерфейса (UI) нового сервиса My.MP3.com. Ему пришлось столкнуться с незавидной (хотя и не такой уж необычной) задачей: разработать особый UI для несуществующего пока сервиса, более того, сервиса, принципы деятельности которого были неизвестны. Эта последняя загвоздка была ключевой: в одной области нашего веб-сайта должны были осуществляться онлайн-запросы к базе данных, а существующие прототипы были медленными. Слово «медленный» в применении к нашему новому, пионерскому, новаторскому, с иголочки сервису (My MP3.com) казалось менеджерам невозможным, и Джеймсу предстояло понять, как создать работоспособный UI, который не требовал бы от баз слишком много данных и тем самым минимизировал бы время отклика на запросы.[1]

Но тут случилось кое-что интересное: Джеймс привлек к работе двух своих коллег из отдела дизайна – Брайана Каллахана и Нэнси Бахман, – и выяснилось, что эти ребята не тянут. Не то чтобы они были

[1] Если мне не изменяет память, были еще какие-то технические проблемы, связанные с совместимостью нашей доморощенной системы XML-шаблонов с данными XML, которые создавались онлайн из запросов в базу данных и не существовали в виде файла в файловой системе, но тут я могу ошибаться.

некомпетентны или не добавляли ничего проекту – вовсе нет, – но они просто не принадлежали к «Лиге крутых ребят» (пока что). Или так можно было подумать из-за косых взглядов, которые бросали на них на совещаниях, куда их приводил Джеймс, или отсутствия приглашений на эти совещания! Это был очень странный феномен, но в MP3.com он проявлялся очень часто: или ты «избранный», и в этом случае ты не можешь совершить ничего неправильного, или ты один из тех зануд, которым поручались задания типа «кто-то это должен делать, но это же не круто». Никакой славы, просто поддерживай сайт, чтобы он работал нормально.

Это оказало определенное воздействие и на процесс управления. Если вы были «в струе», то высокопрофильные проекты доставались вам и, занимаясь ими, вы все больше выдвигались на передний план. В инженерном отделе менеджеры постоянно боролись за позицию для атаки на пост директора или вице-президента, и быть на виду означало подъем.

Конечно, успешно справиться со всеми задачами можно было только благодаря подбору команды. Если у вас были лучшие сотрудники, или коллектив с великолепным командным духом, или какой-нибудь парень-гений, то проект сам шел в руки. Иногда проект давали вашей команде, потому что кто-то из нее обладал самыми подходящими для такой работы качествами, а иногда – потому что вы хорошо справились с предыдущей работой.

Итак, менеджеры постоянно старались заполучить лучших сотрудников, чтобы добиться успеха самим. Любые перестановки в области управления приводили к ситуации, похожей на школьный двор: сначала выбираю я, потом ты, потом опять я. Не то чтобы те, кого выбирали, могли получить психологическую травму, долго ожидая своей очереди, но все же у них оставалось сознание того, что от них отказались ради кого-то более «крутого».

В моральном смысле ситуация собственно для инженеров была двоякой. Были стимулы продолжать хорошо работать – «Если я буду крут, то мне дадут крутые проекты» или «Если я очень постараюсь, то, может быть, мне дадут тот другой проект», – но если вы долгие часы выкладывались, а на вас в итоге наплевали, то удар мог быть страшным: «Я много месяцев работал по 80 часов в неделю, а теперь вы мне предлагаете заняться этой скучной поддержкой? Подите к черту!»

Впрочем, перераспределения происходили нечасто, по крайней мере в инженерной сфере – примерно каждые 4–7 месяцев, и никто не мог понять, чем это кончится. Лихорадка начиналась, когда кто-то из топ-менеджеров покидал компанию или – что было более частым – терял расположение Майкла, и тут же начинался всплеск активности: оставшееся руководство запиралось в конференц-зале, сотрудников отво-

дили в сторонку и спрашивали: «Хочешь работать со мной?» или: «Не хотел бы ты руководить командой инженеров?»; шел настоящий переворот, который можно было не замечать, если не обращать на него внимания, но который был очевиден тем, кто знал, куда смотреть.

Джеймс, вероятно, уязвил чье-то самолюбие, приведя с собой в проект людей «извне» – либо из-за того, что они не входили в группировку «правильного менеджера» и, таким образом, добавляли престижа кому-то другому, либо из-за какого-то неоспоримого элитаризма, – но они втроем сумели все сделать. Собственно, они не просто все сделали, но сайт был замечательным, а с точки зрения конечных пользователей – вообще конфетка. Он работал и выглядел как надо.

Однако с течением времени выяснилось, что самым элегантным решением множества проблем было применение JavaScript. Представляю, как вы удивляетесь, зачем я это говорю. Но вспомните, что шел 1999 год и что, хотя JavaScript давно никого не удивляет, его применение ограничено двумя-тремя основными веб-броузерами.

Ну и что?

А то, что мы вообще не использовали на нашем сайте JavaScript, потому что не могли обеспечить одинаковые условия всем пользователям: разные броузеры поддерживали разные вещи, клиенты просто выключали JavaScript, и вообще творился сущий кошмар, потому что было непонятно, что будет работать, а что нет. Было и так сложно убедиться, что страницы выглядели одинаково при различных броузерах и архитектурах, а добавлять язык скриптов, к тому же не всеми поддерживаемый, было явно не с руки.

Но какие времена – такие и меры, и мы готовили большой скачок вперед с нашим сервисом «для продвинутых», используя JavaScript для оптимальной работы. Повторюсь: красота и мощь такого решения были очевидны. Джеймс вместе с директором по веб-дизайну попытались применить JavaScript и, запустив его, перенесли весь сайт My.MP3.com в новое неведомое королевство.

Сайт был действительно так красив, что наш *арт-директор* принял стратегическое решение: переделать весь веб-сайт так, чтобы он соответствовал дизайну нового сайта My.MP3.com.

Занавес поднимается и открывает инженерный отдел MP3.com. Каждый сидит в своем отсеке. Почти полночь, пятница накануне Рождества. Веб-команда и отдел качества уже более двенадцати часов работают над обновленным веб-сайтом, сообщают об ошибках, отлаживают шаблоны и вообще всячески стараются доделать новый дизайн до конца. Один из веб-дизайнеров, Брайан Каллахан, по-видимому, последнее время спал не больше трех часов в сутки, часто уходя с рабо-

ты в шесть утра. В офис входит руководитель группы. У него не просто затуманены глаза: он отчетливо видит, как его жена подает на развод, особенно сейчас, когда в его квартире в 10 милях отсюда вовсю празднуют Рождество, а он все еще торчит здесь.

Наконец Брайан выкрикивает: «Кто-то может мне объяснить, какого *хрена* нужно все закончить именно сейчас?»

Недоуменные взгляды.

Наконец, Меда Парликар, руководитель отдела качества, подходит к нему, треплет по плечу и говорит: «Да, все мы устали. Возможно, пора по домам».

Занавес. Брайан едет домой, наконец готовый праздновать... или отключиться.

Я такой умный: У–М–Н–Ы–Й

Когда все вернулись с рождественских каникул, компания, по словам Робина, «неслась во весь опор». Все работали безумное количество времени, особенно наши системные администраторы (им, кажется, приходилось создавать с полдюжины новых серверов в день, не забывая о повседневных занятиях вроде установки новых сетевых решений – то есть таких, которые не падали при работе, – и управления терабайтами данных) и отдел качества, который должен был тестировать каждый новый компонент, когда он был закончен и поступал на сайт. Я в это время боролся одновременно с расширениями Perl и модулями Apache, которые были функциональными, но в области сообщений о данных содержали давние ошибки. Для меня это было вроде привычного битья головой об стену: я просто хотел это сделать. Кроме того, я, честно говоря, завидовал Дегенхардту: хотя мы работали над этим параллелизмом вместе, именно ему предстояло его внедрять. Возможно, это было и справедливо: он сделал отличную работу, а у меня и так свободной минутки не было, но все же я чувствовал смутную ревность – зря, что ли, я день за днем сражался с Apache и Perl!

Мы продолжали периодически контактировать, обмениваясь идеями по поводу внедрения (а может, это просто я задавал ему вопросы, чтобы быть в курсе, что вообще происходит!), и одной из лучших его идей было применение образцов MP3-файлов для состояний ошибки. Например, если вы включили датчик безопасности, вам выдается сообщение об ошибке MP3 в вашем списке прослушивания вместо требуемой песни. И это были не какие-то MP3, а пятисекундные отрывки из Гомера Симпсона: например если вы пытаетесь воспроизвести больше одной песни сразу, сервер подает звуковой сигнал, и вы слышите слова Гоме-

ра: «Я такой умный: У-М-Н-Ы-Й!» Блестяще! (Хотя эта фраза в итоге навсегда засела у меня в голове.)

Однако самое смешное случилось, когда пришло время тестировать систему. Измочаленные тестеры пришли к нему и сказали: «Ничего не понимаем. Сервер проигрывает все время одну и ту же песню, и этот парень все время говорит...»

Блин!

Если отвлечься от подобных недоразумений, то отдел двигался верным курсом. Конечно, небольшие обломы бывали: например за неделю до ввода в эксплуатацию мы решили переместить наши серверы на новейший центр данных, но все ведь объяснялось ажиотажем, правда? Правда...

Что ж, признаться, были и другие мелкие трудности. За неделю до назначенного на 11 января запуска сайта Майклу представили систему в тестовом режиме, и он был просто в ярости: «Кто за это отвечает???» В мгновение ока вице-президент по инженерной части, директор по вебдизайну и почти все остальные, кто принимал хоть какие-то решения, были согнаны в офис Майкла. Несложно догадаться, в чем была проблема: *JavaScript*!!! Это была негативная сторона менталитета MP3.com, который поощрял инициативу: все шло прекрасно, пока вы чего-то не сделаете не так или просто не окажетесь с Майклом по разные стороны баррикад. Неприятно, но так и было: работаешь-работаешь, следуя своей музе, ведешь свою игру, паришь в облаках с ощущением своей неуязвимости и избранности – и вот ты... никто. И ничего не поделаешь.[1]

Но несмотря на все проблемы, часть которых проблемами не были, сервис My.MP3.com запустился почти без проблем. Мы открылись после полуночи 12 января 2000 года – через день после предполагаемого срока. Самым серьезным недостатком стало то, что только около половины серверов при первом запуске отображали новый контент, но уже через некоторое время все наладилось.

А компания? Что ж, те, кто присутствовал при запуске – в основном инженеры, – были просто в экстазе. Рекой лилось пиво и шампанское, все стояли неподалеку на тот случай, если что-то произойдет, завороженные открывающимися перспективами.

---

[1] Если разобраться, то это похоже на «Уловку-22»: или добираешься до солнца, но в стиле Икара, или идешь в никуда. Всегда есть риск залететь слишком высоко. Я был среди тех, кто не очень стремился наверх; когда мой менеджер сказал мне: «Мы подумываем назначить тебя на руководящую должность...», – я немедленно ответил: «Кого, меня?» Не самый лучший ответ для того, кто хочет чего-то добиться в жизни.

Совещание в курилке инженерного отдела

Состояние в MP3.com после запуска My.MP3.com можно сравнить разве что с посткоитальным. Все были шокированы – в хорошем смысле – и просто ждали, что будет. Мы просматривали новостные сайты и ждали, что же случится, *в конце-то концов*. Было понятно, что RIAA или кто-то из крупных лейблов вскоре предпримет *действия*, но помимо довольно мелких вызывающих высказываний все шло спокойно, даже как-то черсчур спокойно. И вскоре все вернулось к «нормальному» течению дел, или по крайней мере к режиму «не аврал». Не то чтобы нам было нечем заняться – как раз наоборот. Дел было по-прежнему по горло: My.MP3.com подвергался полной переработке, мы улучшали другие элементы нашей музыкальной инфраструктуры, чтобы они смогли справиться с теми функциями, которые мы разработали для Da Bomb, ну и к тому же мы вернулись к повседневным занятиям, которые пришлось отложить на время разработки Da Bomb.

И это было довольно странно, потому что все чаяния, реклама и энергия, которые ушли на создание My.MP3.com, как будто рассеялись. Больше не было проекта, над которым так бурно хотели работать все, не было ощущения, что «это нечто большее, чем я, большее, чем то, что я обычно делаю», заставлявшего людей работать сверхурочно, лишь бы быть причастными к этому. Нет, мы по-прежнему работали по 60–70 часов в неделю, просто больше не проводили столько часов вместе, не чувствовали себя единым организмом, вернувшись к частным задачам и небольшим группам.

Но когда ход вещей вернулся к нормальному, чехарда в руководстве, которая немного поутихла было, возобновилась. Более того, тогда она впервые коснулась меня.

Сначала прошло совещание, которое мы с моим менеджером пытались назначить последние два месяца и на которое он постоянно не мог прийти (что вполне понятно, поскольку он руководил крупнейшим проектом за всю историю компании). Мы зашли в один из боковых кабинетов и...

«Том, у меня для тебя хорошая новость: парень, ты получил повышение!»

(Диалог продолжается, обсуждаются сумма и дата повышения, я терпеливо жду, что будет дальше.)

«Дэвид, я счастлив, но, честно говоря, это не то, чего я ожидал, когда мы разговаривали с тобой при приеме на работу. Ты подчеркнул тогда, что я могу ожидать заработка от X до Y через три-пять месяцев; прошло шесть месяцев, а новая зарплата не дотягивает и до обговоренного минимума».

«Жаль, что ты разочарован. У тебя есть наш разговор при поступлении на работу в письменной форме?»

(Моральная пощечина.)

«Нет, нет...»

«Что ж, тогда извини, Том. Тогда я ничего не могу поделать...»

Я пришел в раздражение. Я чувствовал, что получал зарплату меньше, чем достоин, но считал, что все будет исправлено в относительно короткое время, как только я докажу, что могу хорошо работать. И разве я плохо работал? Или меня кинули просто потому, что так работают *все* компании: нанимают вас за ничтожную сумму, а потом прибавляют жалованье, но существенно – редко.

Ответ на вопрос, как я работал, был получен через пару недель, когда меня перевели из группы Дэвида к Дженет Дельфино в группу инфраструктуры. Хотя это логически вытекало из проектов, над которыми я в то время работал, выглядело это как понижение в должности. Меня обменяли на одного или двух программистов из отдела Дженет, и в глазах ребят из отдела Дэвида меня просто отправили на низший уровень. Я потерпел неудачу и чувствовал себя неумехой – точнее, больше я не был одним из тех Избранных, которым дают работать над крутыми проектами.

Антракт: создание «Заповедника панд»

Через два месяца после запуска сервиса My.MP3.com настало, наконец, время переводить инженерный отдел в новое здание. Несколько месяцев мы терпеливо ждали, пока Рэнди Блюмхаген, наш менеджер по оборудованию, махал своей волшебной палочкой, и результат нас не разочаровал. Заходя в здание, вы тут же оказывались перед широкой изогнутой лестницей, которая вела на третий этаж (второй по-прежнему был в ремонте и выглядел непрезентабельно), черно-красный ковер с узорами пламени вел вас на 30–60-футовое открытое пространство, уставленное стильной мебелью – мягкие кривые, увеличенные спинки и подлокотники, – все это соответствовало духу склада с высокими потолками, окруженного рядами офисов с каждой стороны. Слева и справа были стулья, кушетки и столы – места общих собраний; стены закруглялись, как в башнях замка, а окна с одной стороны выходили на шоссе 805, а с другой – на такое же «неинженерное» здание. Стоянка была заполнена блестящими машинами.

В целом место больше всего напоминало огромный производственный склад с высокими потолками, который обрамляло с внешней, оконной стороны кольцо офисов. Футах в четырех от внешнего кольца офисов

располагалось нечто вроде ряда из двух-трех грузовых контейнеров, между которыми была чуть ли не половина футбольного поля. Это был еще один ряд офисов, но их подавляло пространство, которое действительно было *огромным*. И стильным: офисы были красивыми, современными, дизайн их варьировался: у некоторых окна были как иллюминаторы, у некоторых – как пещеры.

Я оказался в ряду офисов, который с внешней стороны был отделан полупрозрачной гофрированной пластмассой, сквозь которую нельзя было смотреть, но которая пропускала дневной свет. В офисе было еще двое программистов: Джи-Кей Пэриш Филп и Дэвид Дудас, о котором нужно, конечно, сказать отдельно. Пока текли месяцы работы в отсеке, казалось, что беспокойства и жалобы можно было держать при себе, поскольку «как только инженеры переедут в собственное здание, все изменится» и правила игры будут другими. На все нарушения тишины – от разговоров или негромкой музыки до тех, кто не носил сменной обуви – была одна реакция: повторяли одну и ту же мантру, и люди возвращались (порой с завистью) к своей работе, убежденные, что уже скоро все будет хорошо. Подозреваю, что это оказало свое воздействие и на отделку нового помещения, потому что не уверен, что у нас были бы эти офисы, не жалуйся мы так часто, а ведь менеджеры обязаны ценить свой инженерный отдел. Все стало еще лучше, когда нам предложили самостоятельно выбрать себе соседей по офису и тому подобное.

Но к тому времени, когда мы были уже готовы переехать, некоторые из этих обещаний уже стали забываться: ситуация изменилась, о «проблемах» забыли, а всеми предыдущими заявлениями пренебрегли. Сначала встал вопрос о соседях по офису. Нам долго обещали, что мы сможем работать в одном офисе с тем, с кем хотим, но вдруг поползли слухи, будто менеджеры решили, что все их подчиненные должны сидеть вместе. В этом не было никакой логики, помимо вопросов контроля и предпочтений одного или двух менеджеров, но здесь все же возобладал стиль руководства «пусть инженеры будут счастливы», и Дэвид Макколлум (думаю, это был он) пришел к руководству и сказал: «Ребята этого хотят. Пусть сидят вместе. Для рабочей среды это лучше, чем если все будут работать внутри своих групп». Так и случилось. Не стоит и говорить, что перекрестное опыление пошло всем на пользу, так как различные группы – разработчики приложений, системные администраторы, веб-дизайнеры – стали чуть лучше понимать, над чем работают другие и как функционируют вместе разные элементы.

«Реалисты, вон из нашего офиса!»

Я могу с легкостью сказать, что самые теплые воспоминания от MP3.com остались из-за того, что я работал в одном офисе с Дэвидом и Джи-

Кеем. С самого начала мы решили, что развернем наши столы не к окну, а друг к другу, так что двое из нас сидели перпендикулярно окну – повернись на 90 градусов, и ты все увидишь, – а третий лицом к окну: мы составляли три стороны квадрата, а окно – четвертую. В центре находились джунгли растений. Затем мы принесли с собой музыкальные инструменты: Джи-Кей – гитару, бас и диджериду, Дэйв – гитарный усилитель, а я – усилитель для баса. Мы были опутаны проводами, инструменты висели на крюках на стене, и каждый раз, когда в коде возникал затор и нужно было сжечь несколько нервных клеток, можно было поиграть, а не биться головой о стену.

Совершенно непредсказуемые последствия этого возникли из-за того, что в MP3.com вообще работало множество отличных музыкантов. До сих пор помню, как Рэттлснейк Рэй Мэтьюс, веб-дизайнер, зашел в наш офис, ни слова не говоря подошел к месту, где сидели я и Джи-Кей, снял гитару со стены, включил ее в сеть и выдал потрясающий блюзовый запил на две минуты, потом выключил гитару, повесил ее на место и ушел. Черт. Не-ве-ро-ятно. Кажется, никто из нас до того не слышал, как он играет. Это случалось пару раз в месяц, и это были настоящие минуты отдыха, от которых лица расцветали улыбками. (Оказалось, что Рэй довольно долго был сессионным музыкантом и даже ездил в турне с Мадонной, а сейчас устроился на постоянную работу, чтобы – помимо всего прочего – обеспечить сыну нормальную школу.)

В нашем офисе приветствовалось общение – точнее, споры. Джи-Кей был магистром философии из Техаса, впоследствии ставшим программистом, Дэвид – вегетарианцем и буддистом, который вырос в Детройте в интересных условиях, а я – высококвалифицированным адвокатом дьявола, мастером на все руки и человеком с любопытным подходом к жизни. Темы дискуссий варьировались от природы зависимости до лучшего альбома Beastie Boys («Paul's Boutique» был потрясающим, «Check Your Head» – еще круче и вообще лучшим и более глубоким альбомом), от трактатов по работе мозга с информацией до способности определять свою судьбу и утверждений Дэвида и Джи-Кея, что «реальность» – это просто функция субъективного восприятия (которая, следовательно, не существует). Все это обсуждалось с сумасшедшей энергией, и если, случалось, мимо окна пролетал пластиковый пакет, то это служило началом дискуссии о символизме пластикового пакета в только что вышедшем фильме «Красота по-американски».

Но дело не сводилось к «философским диспутам об истинной природе пролетариата в американском обществе». Мы с Дэвидом регулярно проверяли друг у друга код Perl или просто обеспечивали друг другу лишнюю пару глаз и ушей для резонаторов. Часто бывало так, что энергии, потраченной на *описание* проблемы, хватило бы, чтобы зажечь элек-

трическую лампочку и решить все самому, но ведь это была составная часть процесса! Все это обеспечивало здоровую рабочую среду, где 75% времени сидишь и молча работаешь – либо с наушниками, либо кто-то без вопросов выбирал следующий альбом, – а все остальное время можно было просить совета в случае проблем... но так и ждешь, что проблемы *действительно* появятся.

Подчиненных Джи-Кея мне реально было жалко. Примерно раз в неделю кто-нибудь приходил и вынужден был терпеливо ждать, пока мы прервем нашу причудливую беседу, иногда минут 10 – или уходил, если его вопрос был не очень важен. Самое странное, что редко кто-то вмешивался в наш разговор, чего я никогда не мог понять.

Вероятно, в основном наш бред приходилось выносить Брайану Каллахану, поскольку он часто проходил мимо нашего офиса. Когда он, наконец, ушел из MP3.com, чтобы самому поискать счастья, то опубликовал на внутренней электронной доске объявлений длинный список лучших воспоминаний и других мелочей. О нашем офисе он сказал: «Предлагаю всем, кто считает себя реалистами, зайти в офис к Дэвиду и Джи-Кею» (я в то время уже куда-то переехал), на что Дэвид тут же ответил:

«Реалисты, вон из нашего офиса!»

Точно сказано.

Не взрыв, а всхлип

Еще до переезда в новое здание на нас подали в суд за нарушение авторских прав RIAA (от имени крупнейших звукозаписывающих лейблов), агентство Гарри Фокса, которое владело правами издания на большую часть музыки из нашей «библиотеки», и несколько более мелких лейблов.[1] Это произошло 21 января, через 10 дней после запуска My.MP3.com. Хотя в этом не было ничего неожиданного, но беспокойство ощущалось, особенно из-за языка некоторых газетных статей: «явный случай нарушения авторских прав...» и так далее. Проходили особые коллективные совещания, на которых нам говорили, что все нормально, в соответствии с нашими ожиданиями. На еженедельных собраниях нам рассказывали новости и заверяли, что мы все делаем правильно, что все в порядке и что у нас достаточно денег в банке, но все же ситуация была тревожной. Однако мы не прекращали конвертировать компакт-диски: наоборот, каждый день в базу добавлялось по 1500 альбомов.

[1] У песен есть два типа авторских прав: права издания и права записи. На каждый тип выдается отдельная лицензия.

В конечном счете, я не могу припомнить, чтобы велись открытые разговоры о прекращении нашей деятельности. Не знаю, связано ли это с тем, что все твердо верили в победу (собственно, это было в их интересах: кто хочет ходить мрачным, когда риск может привести к богатству), или просто никто не хотел это обсуждать. Честно говоря, мы чувствовали себя отчасти неуязвимыми: подобралась великая команда инженеров, мы решали все проблемы на своем пути, доходы лились рекой... казалось, нам ничто не страшно. Мы изменяли мир. Было ясно, что мы сделали следующий шаг в музыке, что лейблы морально устарели и рано или поздно поймут, что музыка должна распространяться в цифровом формате, что DRM-технология потерпела крах навсегда – и вот появляемся мы, с великолепно сделанным сервисом, с блестящими статистическими отчетами, из которых *точно* известно, сколько раз, кем и какая песня проигрывалась. Возможности работы с данными были безграничны, возможности вообще – тем более, да и вообще это даже привлечет им новых потребителей! Собственно, мы хотели заплатить им за лицензирование. Мы уже собирались платить артистам напрямую в зависимости от количества прослушанных песен, чтобы они получили те роялти, какие заслуживают. Что могло пойти не так?

Но шло время, суды назначались и проходили, и оптимизм тоже затухал. Из компании начали массово уходить сотрудники. Уходили по разным причинам: кто-то зашел в тупик, потому что попал в опалу и больше не мог работать ни над чем интересным, а просто тратил время (при этом снижалась производительность), другим не нравилось направление, в котором двигалась компания: теперь нашей целью было «Делай деньги, поддерживай и улучшай инфраструктуру!», а не «Изменяй мир!», что казалось таким волнующим в начале пути. Конечно, My.MP3.com и был попыткой изменить мир, но он был готов, и теперь нужно было просто ждать, выйдет ли он на земную орбиту или провалится. Это был наш главный удар по миру, а остальные ресурсы были направлены на получение доходов, что с изменением мира никак связано не было.

Потом произошли первые сокращения среди инженерного состава. В других отделах уже были увольнения, но нас это пока не затрагивало, мы были в безопасности. Я узнал, что парни из другого здания называли инженерный корпус «Заповедником панд»: место, где редкий вид получал соответствующий уход и кормежку и ограждался от всех опасностей реального мира. Все. Стекло разбилось вдребезги.

Собственно, некоторые (или все?) уволенные первыми стали почему-то хуже работать – низкий боевой дух, истощение; может быть, им просто давали меньше шансов проявить себя, что выразилось в общем упадке производительности. Трудно было перейти от ощущения «на вершине

мира» к «мы даем тебе черную работу, потому что ты кого-то достал»; трудно было выдерживать тот лихорадочный темп, который был задан, особенно когда тебя постоянно хотят опередить и занять твое место – либо кто-то из компании, либо претендент, мечтающий поработать в MP3.com. Мой хороший друг и бывший сосед по отсеку Джеймс Парк ушел, видимо, поэтому. После выхода My.MP3 он так и не стал прежним: по каким-то причинам его постепенно перевели в статус сотрудника второго класса. Возможно, это из-за JavaScript, или из-за пренебрежения к иерархии и того, что он приводил на совещания своих людей, или просто из-за того, что на праздник по случаю запуска сервиса он только заглянул, а не оставался на несколько часов, чтобы провести их с менеджерами и инженерами, которые создавали сайт. Больше ему шанса показать себя не предоставили.

Самому мне по-прежнему нравилось работать в MP3.com, но я начинал задумываться о дальнейшем направлении своей карьеры. Я поступал в MP3.com, чтобы работать с другими фанатами компьютера, искал такое окружение, которое бы мне подошло и в конечном счете помогло бы повысить мой собственный уровень через сотрудничество с другими. Еще одной причиной была возможность изменить мир. За три месяца до собеседования в MP3.com я обсуждал перспективы музыки в цифровом веке с коллегой-путешественником из Нидерландов. И здесь у меня была возможность непосредственно повлиять на эти перспективы.

Но с изменением направления компании, когда мы перестали работать главным образом на сервис My.MP3.com – или, по крайней мере, когда его юридический статус стал выглядеть неопределенно, – и когда мотивы высшего руководства стали нам непонятны (впервые пошли слухи о «разделе кассы» и о том, что «нас продадут, а сами станут миллионерами»), я начал задумываться о том, что я тут делаю.

Сначала я подумал, что надо было действительно становиться инженером-химиком (у меня было высшее образование, но я никогда не работал по специальности: в Боулдере для химиков было не так много вакансий). Это ощущение усиливалось из-за того, что «академически» я значительно уступал многим коллегам по работе: конечно, я ходил на компьютерные курсы, но теоретической глубины по сравнению с ними мне наверняка не хватало. Я был способен делать свою работу, но тут подумал: ведь у меня *были* такие знания в химии, так что почему бы не пойти по этому пути, где можно было применить все свои ресурсы? Эти мысли усугублялись чувством, что «если работа программиста в доткоме в окружении фантастических парней не для меня, то я потерян для компьютерной индустрии», и так начались сомнения: пока ничего серьезного, но зерно уже посеяно.

Тем временем наши правовые битвы не утихали. Мы договорились с двумя, а потом с тремя крупными лейблами, но в остальных случаях ничего хорошего не светило. Обвиняли судью процесса, еженедельные совещания изобиловали оптимистической риторикой, но Робин выглядел усталым, и на многих речи руководства перестали производить впечатление. Так ли это было? Сложно сказать, каким было общее настроение, потому что все по-прежнему были очень заняты: работы было предостаточно. Но эмоциональный настрой куда-то улетучился. Никто не рвался работать над проектами: все просто исполняли свою работу. Работали по-прежнему по 60-80 часов в неделю, но таким образом не шли к новым горизонтам, а удерживали старые позиции.

Решающий поворот в моем сознании произошел во время работы над краткосрочным – четыре недели – проектом для выхода на рынок; это был наш первый сервис подписки. Работать над ним было одно удовольствие (мне всегда нравились проекты, сжатые по срокам), но отсутствие взаимопонимания между отделом качества и отделом инфраструктуры (то есть мной) было просто убийственным. Например, два или три дня в неделю я в 4.30 утра отсылал по электронной почте письмо: «Вот что я сделал. Вот это должно работать. Вот это не работает. Я поехал домой спать. Буду в 10 утра». И через три часа мой менеджер звонила мне и говорила, как что-то не работает. Тут было две проблемы: 1) я отчаянно трудился над тем, чтобы все заработало, и звонить мне в 8 утра – вряд ли лучший способ решения проблемы; 2) довольно часто отдел качества, просто не прочитав мое сообщение, жаловался на то, что, как я им сам сказал, *уже не работало*. Блин!

Моего менеджера я не обвиняю. Я знаю, что это ей говорил вице-президент, которому директор отдела качества жаловался, что наша программа не работает.[1] Но мне стали сниться кошмары, как у меня звонит сотовый; я вкладывал в проект душу и сердце, а за это не получал даже уважения. Вскоре после запуска проекта (точно по расписанию) я охладел к работе. Да, за свою работу я получил пакет опционов (и не просто символических), но это не могло вернуть мне прежний образ мыслей или чувство, что тебя уважают за твою работу. В этом плане со мной было покончено.

---

[1] От всего сердца могу сказать, что в области защиты своих сотрудников Дженет была неподражаема: она всегда готова была биться за нас, и это было круто. Она не терпела пустой болтовни и всегда шла прямо к подчиненному, чтобы убедиться, что все будет сделано нормально и что вы преисполнены того чувства ответственности, которого порой недостает инженерам. Гм! Но мне этот стиль руководства не подходил: я плохо работаю, когда мне буквально заглядывают через плечо.

Я работал в MP3.com еще год, и дела пошли лучше: один из вице-президентов – Дэн О'Нил – взял меня под свое крыло и стал работать со мной, чтобы моя производительность вернулась на прежний уровень. Он потратил много времени и энергии, и отчасти это принесло результаты. Я выполнял свою работу и писал код, но не вкладывал в труд душу. Я работал по 40-50 часов в неделю, что в такой среде было просто лодырничаньем. На следующий день после 11 сентября я уволился.

Эпилог

28 апреля 2000 года предварительным решением MP3.com была признана виновной в нарушении авторских прав; итоговое судебное заседание было назначено на август. Подверженный «пространственной манипуляции» контент сервиса My.MP3.com вскоре был закрыт, и на ваших личных страницах My.MP3.com появились значки, напоминающие висячий замок и свидетельствующие, что данная песня недоступна для прослушивания. MP3.com сумела договориться с четырьмя из пяти ведущих лейблов, согласившись выплатить отступные и приобрести лицензию на их контент после установления авторских прав.[1]

28 августа 2000 года MP3.com судилась с последним лейблом, который не пошел на урегулирование, и девять дней спустя проиграла суд. Еще до выплаты ущерба было достигнуто соглашение, но все уже было потеряно.

В последующие месяцы MP3.com держалась стойко, разыскивая новые источники дохода, партнеров и «революционные» приложения. Они возвращали на сайт лицензированный контент, висячие замки стали исчезать со страниц, но доступ был открыт только к 3–4 песням из 10 на альбоме (разочаровывающе, ничего не скажешь). Через полгода после договора с последним лейблом MP3.com была именно этим лейблом – Vivendi Universal – и приобретена.

[1] Этот процесс «подтверждения авторских прав» затянулся на многие месяцы, потому что лейблы – которые жили далеко не в цифровом веке – отвечали на запросы о подтверждении очень медленно. Не лучше вело себя и агентство Гарри Фокса, и как-то раз мы послали им две или больше коробок, до краев заполненные формами запросов подтверждения авторских прав (на каждый альбом требовалось индивидуальное подтверждение). Сколько человеко-часов это отняло у MP3.com, мне трудно даже представить, но я бы все отдал за возможность посмотреть на лица ребят из агентства в этот момент.

ХРОНОЛОГИЯ СОБЫТИЙ

1. 12 января 2000 года: взрыв Da Bomb.
2. 21 января 2000 года: RIAA подает на MP3.com в суд.
3. 28 апреля 2000 года: MP3.com проигрывает предварительные слушания по делу нарушения авторских прав; итоговые слушания назначены на август.
4. 4 мая 2000 года: Контент крупнейших лейблов на сервисе My.MP3.com «закрыт» для использования, но по-прежнему фигурирует в личных кабинетах пользователей.
5. 9 июня 2000 года: MP3.com заключает соглашение с Warner Music Group и BMG Entertainment, получая лицензию на их контент для My.MP3.com.
6. 28 июля 2000 года: MP3.com заключает соглашение с EMI, получая лицензию на их контент для My.MP3.com.
7. 22 августа 2000 года: MP3.com заключает соглашение с Sony Music Entertainment, получая лицензию на их контент для My.MP3.com.
8. 28 августа 2000 года: дело MP3.com поступает в суд: «UMG против MP3.com».
9. 6 сентября 2000 года: MP3.com проигрывает дело; судья Ракофф принимает решение о выплате убытков в пользу Universal Music Group.
10. 18 октября 2000 года: MP3.com заключает соглашение с агентством Гарри Фокса, регулируя вопрос о нарушении издательских авторских прав.
11. 15 ноября 2000 года: MP3.com заключает соглашение с Universal Music Group, получая лицензию на их контент для My.MP3.com.
12. 21 декабря 2000 года: MP3.com вступает в переговоры по лицензированию Warner Music.
13. 21 мая 2001 года: Vivendi/Universal объявляет о своем намерении приобрести MP3.com.

Оглядываясь назад, можно сказать, что продолжать конвертировать компакт-диски в то время, когда у нас явно не хватало на это денег, было очевидной глупостью, но это решение было полностью в духе компании. Видите ли, в какой-то момент MP3 стали «плохишами» интернет-

пространства – может быть, только в нашем собственном понимании. Мы столкнулись нос к носу с крупнейшими звукозаписывающими лейблами и не отступили: более того, мы подали на них *встречный иск*, после того как они подали на нас в суд за нарушение авторских прав!

И все это было благодаря Майклу. Он определял собой MP3.com и сделал ее такой компанией, которой она стала: компанией, в которую стремились таланты, потому что просто хотели там работать. Подобная культура, заданная раз навсегда, и удерживала компанию вместе, и разрывала ее на куски.

Прошло уже больше восьми лет после взрыва Da Bomb, а вопрос скачивания музыки и обмена ею продолжает подниматься. Некоторые цели сервиса My.MP3.com были достигнуты: Apple открыла музыкальный магазин «iTunes» и стала продавать музыку, – а некоторые оказались неуместными или еще ждут своего часа. В целом же просто поразительно, сколько еще дискуссионных моментов...

Послесловие

О своей работе в MP3.com я могу много чего порассказать, гораздо больше, чем сумел втиснуть в эту главу. Я работал там в течение всего двух бурных лет, но никогда не забуду это время.

Среди самых удивительных моментов, так или иначе, можно отметить длительность дружеских отношений, которые я завел в эти два года. Я общаюсь с большим количеством ребят из MP3.com, чем с любого другого бывшего места работы. И не только я. Разговаривая с прежними коллегами, чтобы проверить даты и услышать их версию событий, я неизменно слышал: как удивительно, сколько людей оттуда все еще общаются между собой и какие связи сохраняются между сотрудниками даже через пять-шесть лет после их ухода. Не уверен, что все это достигалось совершенно сознательно, через прием на работу только «правильных» кандидатов, способных пережить данный опыт, но так или иначе это сработало. Многие сотрудники до сих пор работают вместе, хотя все они согласны в одном: так, как в MP3.com, больше никогда не будет.

8

Inner Source – ПО с исходным кодом, открытым для внутреннего использования

Интервью с Ауке Джилдердой

Нас всегда поражало, что самые большие и успешные проекты свободного ПО, вроде Linux, Firefox и Apache, могут выпускать качественные программные продукты в условиях, где большинство команд просто сгорело бы. Это крупные (а иногда и огромные) коллективы, разбросанные по всему миру, и они во многом зависят от работы, выполняемой бесплатно или за смешные деньги добровольцами: программистами, тестировщиками и другими специалистами. Поэтому нам захотелось пообщаться с Ауке Джилдердой, который изучал работу команд разработчиков свободного ПО и успешно применял принципы этой работы в корпоративных проектах.

Дженни: **Вы посвятили немало времени исследованиям проектов ПО с открытым исходным кодом и определению принципов их работы, применимых в работе компаний. С чего все это началось?**

Ауке: Моя работа начиналась в исследовательском департаменте крупной международной корпорации, базировавшейся в Европе. Я работал в группе архитекторов ПО в одной из лабораторий, мы занимались разными аспектами архитектуры программного обеспечения.

Я заметил, что в мире ПО с открытым исходным кодом есть целый ряд впечатляющих успехов, но при этом есть и целый ряд полных провалов. Например, Apache обошел крупных игроков – IBM и Microsoft – на рынке веб-серверов, хотя те приложили немало усилий для победы, пытаясь получить контроль над веб-стандартами. В то же время подавляющее большинство проектов на сайтах по разработке ПО с открытым исходным кодом находятся в подвисшем состоянии, фактически представляя собой чью-то инициативу, которая так и не дала результатов. Иными словами, у свободного ПО есть чему поучиться, причем весьма серьезным вещам.

Компания, в которой я работал, на тот момент была, да и сейчас является крупным специалистом в области программной архитектуры для программных семейств продуктов, но она многому могла научиться у сообществ разработчиков свободного ПО в плане организации процессов разработки, в частности, тому, как наладить совместную работу невзирая на организационные препоны. Я изучал вопрос разработки ПО с открытым исходным кодом и работал над внедрением ключевых аспектов этой разработки в компании. После я перешел в подразделение программных продуктов, мы взяли некоторые мои идеи по поводу ПО с исходным кодом, открытым для внутреннего пользования (inner Source – по аналогии с open source, ПО с открытым исходным кодом) и создали новую среду разработки, CollabNet, где и применили эти идеи. После двух лет такой работы я был готов к новым задачам и перешел в CollabNet, чтобы применять эти идеи и в других организациях.

Эндрю: **Не могли бы вы рассказать что-нибудь об одном из проектов, которые вы изучали? Может быть, о проекте, где приходилось решать проблемы, связанные с командой?**

Ауке: Например, я довольно плотно изучал работу проекта KMail, это программа для чтения электронной почты при менеджере рабочего стола KDE. Очень впечатляющий проект, работало на нем человек двадцать, люди весьма неординарные, между которыми время от времени случались конфликты. Скажем, один хотел возглавить проект, а у других было иное мнение на этот счет. Интересно было наблюдать, как они находят выход из такой ситуации.

Интересно было также наблюдать за тем, как у них был организован сам процесс разработки. На каком-то этапе они начали объединять отдельные элементы работы. У них была идея дать пользователям возможность шифровать сообщения. Они взяли стандартное шифрование и создали интерфейс, к которому можно было его привязать. На этом этапе я увидел, как эта команда взаимодействовала с другими командами. И у них были разные способы сотрудничества.

Я изучал проекты извне, не участвуя в работе этих команд. И собрал ряд наблюдений по ключевым, на мой взгляд, аспектам, определяющим успех проекта свободного ПО. Следующим шагом было, конечно, внедрение этих аспектов в реальную среду разработки на предприятии.

Дженни: **Похоже, вы не только изучали вопросы разработки продукта, но и наблюдали за тем, как люди взаимодействовали в этой работе. Чем, по-вашему, объясняется сложившийся способ взаимодействия в командах разработчиков ПО с открытым исходным кодом?**

Ауке: В командах, работающих над ПО с открытым исходным кодом, постоянно видишь, как люди проходят определенные стадии развития. Обычно они начинают не как разработчики свободного ПО. Девять из десяти человек начинают как пользователи существующих программ, в какой-то момент они начинают вносить туда какие-то поправки, становятся опытными пользователями, затем некоторые из них начинают как-то помогать этому проекту и в итоге проходят к тому, что становятся разработчиками. В успешных проектах можно видеть, как грамотно выстроено отношение к людям, чтобы поддерживать и стимулировать такой путь развития. Например, в успешных проектах свободного ПО всегда очень просто направлять отчеты о недостатках и неисправностях. Они делают данную процедуру очень простой – помогают людям, столкнувшимся с проблемой, не жалея времени на то, чтобы разобраться, в чем эта проблема заключается. Во-первых, потому что это в их интересах, а во-вторых, потому что это стимулирует развитие пользователя, от начального уровня до экспертного и затем до уровня разработчика.

Дженни: **Это очень интересно, потому что очень многие работают в проектах разработки ПО, плохо разбираясь в деталях вопроса, который они решают. Значит, одним из факторов успешности проектов свободного ПО является то, что эти люди хорошо знакомы с продуктом с точки зрения пользователя и только потом приходят к тому, чтобы вносить в него изменения?**

Ауке: Да. В большинстве случаев я видел, что люди начинают с пользования продуктом. Затем они начинают содействовать и сотрудничать, а потом переходят в разряд разработчиков. Ну и, конечно, на каждом

этапе совсем разные порядки цифр. Обычно разработчики исчисляются десятками, участники проекта – сотнями, если не тысячами, а счет простым пользователям идет на десятки или даже сотни тысяч.

Эндрю: **Допустим, мы имеем ситуацию с работой в компании, где у разработчиков нет возможности начать с уровня обычного пользователя. Что можно сделать, чтобы каким-то образом применить тот принцип, о котором вы сейчас говорили? Очевидно, именно он представляет фундаментальное отличие большинства проектов свободного ПО и тех проектов, где профессиональные разработчики получают задание от компании.**

Ауке: Здесь два ключевых аспекта, и в компаниях мы имеем то же самое. Один аспект в проектах свободного ПО – собственно, проектирование и разработка, то есть ключевые принципы открытого способа разработки. Во-первых, это сама открытость процесса, и есть ряд шагов, которые вы можете предпринять, чтобы сделать процесс разработки открытым. Первый – это, конечно, обеспечение доступности информации по ведущимся разработкам, а в компаниях чаще всего вся ведущаяся работа по умолчанию закрыта для доступа. То есть вы ничего не знаете о проектах, над которыми работают другие, даже если работаете в одной компании, а то и в одном отделе. Вы получаете доступ к информации только в случае возникновения служебной необходимости и после прохождения разных специальных процедур для получения такого доступа.

Если мы в одной команде, и вам нужно применить компонент, над которым я работаю, наверняка вам понадобится модифицировать его под свои нужды. Чтобы облегчить вам этот процесс, я мог бы в постоянном режиме выкладывать промежуточные версии того, что я создал, проверил и каким-то образом протестировал, чтобы вам и другим пользователям не пришлось тратить слишком много времени на то, чтобы разом вникать во все изменения. Иными словами, я делаю разработку более открытой за счет того, что облегчаю доступ к своей работе. И тем самым стимулирую многих начать использовать мой продукт.

Это одна сторона, сам аспект разработки ПО. С другой стороны, мы имеем, если так можно сказать, «принципы общественного управления». И нужно очень грамотно выстроить эти два аспекта, чтобы получить успешный проект свободного ПО. В девяти из десяти компаний есть только один аспект – разработка, а «общественное управление» заменяется прямым руководством.

Эндрю: **Давайте представим, что наши читатели сейчас подумали: «Надо же, именно в такой «закрытой» компании я и работаю! А хотелось бы, чтобы она стала ближе к модели проектов ПО с открытым ис-**

ходным кодом». Но они понятия не имеют, с чего начать. Что бы вы им посоветовали? Какие нужны шаги, чтобы перейти из точки А в точку Б?

Ауке: Компания, в которой я работал, может послужить хорошим примером. Все остается в том же виде: решения о распределении задач, управление проектом, стратегия разработки, планы действий, управление требованиями, обязательства перед пользователями. Но сам подход к разработке фундаментально меняется.

Есть несколько необходимых шагов. Я выделил три ключевых аспекта. Первый – открытость, или свобода доступа ко всей информации, имеющей отношение к разработке. Включая исходный код, отслеживание изменений, документацию и взаимодействие разработчиков. Кроме того, этот доступ должен быть максимально упрощен. Пусть информация хранится в близком и доступном месте и для ее получения не надо проходить огонь, воду и медные трубы.

Второе – мы ввели четкие границы прав собственности и контроля: компонент принадлежит команде, разрабатывающей и поддерживающей его, а команда, модифицирующая его под свои нужды, является владельцем «патча», его модификации. Это обычное дело для проектов свободного ПО, но в проектах разработки в рамках компаний такие права и контроль над отдельными компонентами обычно не имеют четкого определения, особенно в пограничных случаях сотрудничества двух команд.

Третье – когда обеспечен необходимый доступ к информации и ресурсам и четко определены права собственности и контроля, мы поощряем команды к тому, чтобы они начали адаптировать или модифицировать компоненты, разрабатываемые другими командами, вместо того чтобы пытаться к ним приспособиться и писать связующие коды. Вот три аспекта в технической разработке, которые можно начать внедрять.

В нашей компании было около 120 разработчиков в двух разных офисах. Основной их задачей была разработка большого количества компонентов, которые затем использовались в нескольких продуктах. И было еще 350 человек, работавших в командах по программным продуктам, которые фактически создавали уже конечные программы для пользования. Новые идеи мы начали применять с постепенного внедрения ключевых аспектов: обеспечения открытого доступа ко всем элементам разработки, четкого определения собственности и контроля и поощрения «модификационного» подхода. Мы постоянно проповедовали эти новые заветы среди работников. Процесс должен быть постепенным и эволюционным; прежде чем две команды начинают применять новую модель сотрудничества, их учат в ней работать и постепенно открывают все больший доступ к информации. Затем эти команды становятся примером для других.

Эндрю: **И какова была их первая реакция на это?**

Ауке: Тут был задействован еще и политический аспект. Это семейство продуктов было в каком-то смысле директивой исполнительного руководства, которое объявило, что для нашего подразделения это будет полезно, и нам пришлось внедрять это принудительно. Чтобы добиться чего-то, всегда нужна определенная степень принуждения. Но платить за это приходится тем, что люди в командах – те, кого начали заставлять использовать общие компоненты, разработанные другими командами, вместо собственных разработок, – недовольны тем, что им навязывают такое решение. Они хотели бы иметь возможность решать самим. Но, конечно, имей они такую возможность, они никогда не приняли бы такое решение. То есть тут всегда есть некоторая проблема причины и следствия, курицы и яйца: приходится применять принуждение, а это вызывает негативную реакцию.

Поэтому, когда мы только начали, реакция была отрицательной, в основном потому, что команды специалистов по программным продуктам не доверяли командам специалистов по семействам программных продуктов.

Дженни: **То есть, когда вы начали, между командами уже были напряженные отношения?**

Ауке: Да. Например, у подразделения по семействам ПО были определенные обязательства перед командами, занимающимися конечными продуктами. И когда я пришел к последним и начал им рассказывать, хотя я был посторонним человеком – то есть не относился и к группе по семействам ПО, – их первая рефлективная реакция была: «Так, „семейства“ вводят этот новый открытый метод, чтобы перестать оказывать нам поддержку. То есть они откажутся от обязательств, и нам придется самим все отлаживать».

Эндрю: **И как вы их убедили? Как вы справились с этой проблемой и преодолели сопротивление?**

Ауке: Пришлось очень много общаться и объяснять, как работает эта модель. Все обязательства оставались неизменными во всех отношениях. В этой ситуации команды, которые строили взаимодействие в течение многих лет в «закрытой» модели, принимали изменения очень медленно. Но были и другие люди, которые только недавно пришли в группу по семейству продуктов, и у них не было этого багажа отношений за спиной. Мы поняли, что это хорошая отправная точка, чтобы они сразу начали работать с семействами продуктов в новом, «открытом» стиле. То есть кто-то должен был пройти это, продемонстрировать на своем примере другим, а потом они смогли бы понять, как это будет работать и в других командах.

Первоначально группы по конечным продуктам не участвовали активно, а только наблюдали за нашей инициативой и ходом ее реализации. Основной задачей начального этапа было постепенное раскрытие всех элементов разработки. Сначала мы сделали доступным для всего коллектива исходный код в режиме «только чтение». Затем мы организовали рассылку, где каждый мог задавать вопросы, при этом вся переписка была доступна всем. До этого момента всем приходилось задавать свои вопросы «группе поддержки». У этой группы была непростая задача, они находились между молотом и наковальней. Получали вопросы, с которыми шли искать кого-нибудь из группы по платформе – а это 120 человек, – и среди них надо найти нужного специалиста, получить от него ответ и передать спрашивающему. То есть они были посредниками, и, как обычно, в связи с «трудностями перевода» многие детали пропадали при передаче. Ну и в результате, конечно, поддержка работала малоэффективно.

Главное, чему может научиться компания-разработчик у проектов свободного ПО, это то, что в большинстве случаев нельзя использовать разработанный другими компонент без его модификации под свои нужды. В девяти случаях из десяти эта модификация необходима. Так было и у нас. Общие компоненты могут подходить в 80% случаев, но никогда не в 100%. Возникало много проблем. Если были вопросы, обращаться приходилось в группу поддержки, у которой не было готовых ответов. Нужно было выяснять ответ, иногда концы найти не удавалось. Группы по конечным продуктам постоянно испытывали стресс. Получив набор элементов, они должны были запустить их в работу без должного уровня поддержки.

Дженни: **Но ведь недостаточно просто организовать рассылку! У многих компаний есть такие рассылки, и они не очень хорошо работают.**

Ауке: Организационно у группы поддержки оставались те же обязанности и после того, как появились рассылки. Но я включил в список рассылки целый ряд разработчиков более высокого уровня, и они начали отвечать на вопросы. И не успели мы оглянуться, как необходимость в группе поддержки отпала. В принципе, они могли заняться другим делом – разработкой, например, вместо того чтобы искать ответы на вопросы. Больше не было посредничества. Все получили возможность общаться напрямую. Группа по конечным продуктам задавала вопрос и получала ответ на него гораздо быстрее, и ответ был более полным и грамотным, потому что ничего не терялось при передаче через третьи руки, что всегда менее эффективно. В свою очередь, в значительной мере исчезло недоверие и недопонимание между командами. Кроме того, наша инициатива по внедрению такого метода «внутренних ис-

точников» приобрела авторитет, потому что специалисты увидели ощутимые улучшения, явно связанные с новой моделью.

Проблема всех попыток организации общего сотрудничества в крупных компаниях именно в том, что оно слишком общее, не привязано к конкретным областям. Это неизбежно ведет к тому, что энтузиасты начинают публиковать тонны информации, но без четкой цели, и найти ее непросто. Разница нашего подхода заключалась в том, что оно было направлено на конкретный набор общих компонентов, их разработку и поддержание. Весь коллектив сообща использует, адаптирует или разрабатывает эти компоненты. И эта четкая направленность очень меняет дело. Хорошо, когда есть возможность общаться, но необходим и предмет для такого разговора.

Эндрю: **Здорово. Когда вы задействовали главных разработчиков, группу поддержки стало можно перевести на более важную, настоящую работу, что было хорошо и для них, и для компании в целом.**

Ауке: Да. Эффект был прямым и значительным. Это очень простое действие. Совсем не сложно организовать почтовые рассылки и открыть исходный код для общего доступа. Но это дало большой эффект и сильно укрепило доверие.

Тем временем я продолжал свою подвижническую работу, раз за разом проговаривая ключевые аспекты и пользу, которые они могли принести моему собеседнику. Кроме того, я придумал своему подходу название – Inner Source, По с исходным кодом, открытым для внутреннего пользования, – и оно быстро прижилось. Обычно когда люди слышат об этом впервые, они сначала не понимают, о чем речь. Слышат термин и просто повторяют его, не вникая в то, что другие под ним понимают. Но название делает свое дело, они начинают говорить об этом, в какой-то момент начинают понимать и то, о чем идет речь. Важно придумать название для любого подхода.

Дженни: **Почему вы начали проект Inner Source?**

Ауке: Я начал его, потому что видел, что у проектов ПО с открытым исходным кодом есть случаи грандиозного успеха и огромное количество провалов. Другими словами, у них есть чему поучиться, в том числе на их ошибках.

Есть три ключевых аспекта, в которых, по моему мнению, заключается разница между успешными и неуспешными проектами. Первый – это простота доступа и открытость: вся информация, касающаяся разработки, находится в прямом доступе, то есть для общения используются открытые рассылки, исходный код и изменения в нем доступны для просмотра, а также постоянно выкладываются промежуточные версии

разработки. Таким образом, другим проще адаптировать и модифицировать ваш компонент. Второе – это собственность и контроль, четкое определение принадлежности компонента и его модификаций и управления ими: нужно определить, кто за что отвечает. Владелец компонента контролирует создание этого компонента. Команда, адаптирующая этот компонент, контролирует свои модификации к нему и решает, нужно ли предлагать их владельцу компонента, а владелец компонента решает, принимает ли он эти изменения, тем самым осуществляя передачу права собственности на модификацию. А третье – это то, что проекты свободного ПО по умолчанию работают путем создания «патчей», модификаций, а не пытаются искать обходные пути.

Эндрю: **Это очень отличается от того, как работает большинство компаний. И вам это удалось?**

Ауке: Чтобы применить данный подход, я параллельно делал три вещи. Первое, я постепенно открывал доступ к ресурсам разработки. Я начал с организации рассылок, затем сделал доступным исходный код, потом организовал предоставление промежуточных результатов разработки и так далее. Во-вторых, я вел постоянную, неустанную работу проповедника: регулярно приходил к ключевым специалистам разных команд, снова и снова рассказывая им, как должен работать новый подход и что они получат в результате, и прося у них «обратной связи». И третье: я активно поддерживал первые команды, которые занимались адаптацией или модификацией компонентов других команд.

Дополнительной задачей, которая сейчас стоит перед многими предприятиями, была необходимость вытеснения существующей инфраструктуры разработки инфраструктурой нового поколения, допускающей сотрудничество. Нам нужно было заменить предыдущую инфраструктуру IBM Rational на следующее поколение CollabNet, что позволило бы всем разработчикам взаимодействовать вне зависимости от организационных препон и географической удаленности, простым и надежным способом. Инфраструктуры предыдущих поколений затрудняли сотрудничество (и это понятно – они восходят к тем временам, когда еще не существовало такого требования и они не были спроектированы для упрощения этого процесса).

Эндрю: **И вы придумали название Inner Source?**

Ауке: Это оказалось очень эффективным способом. Я провел огромное количество внутренних конференций в компании, и на каждой из них – была ли это встреча только исследовательского отдела или всей компании – я рассказывал о «ПО с исходным кодом, открытым для внутреннего использования», и постепенно в компании заговорили об этом. Поскольку компания очень большая, потребовалось немало рабо-

ты, чтобы вызвать эти разговоры и получить очевидную реакцию. И это хорошо сработало. Конечно, я говорил об этом и при любом удобном случае, в особенности тем командам по конечным продуктам, которые были задействованы в программе по семейству продуктов. Потребовалось около года, чтобы начали появляться результаты. Они заметили положительное влияние моего подхода. А потом начали относиться к нему всерьез.

Тем временем мы делали все процессы более открытыми. Сначала мы открыли доступ ко всем исходным кодам, но нужны были и следующие шаги. Мы сделали общение открытым – рассылка стала включать не только группу поддержки, но и самих разработчиков. Мы сделали информацию по внесению изменений гораздо более открытой. И мы ввели так называемые промежуточные версии разработки. Тут пришлось вести жаркие бои с командой по семейству ПО, потому что предоставлять их еженедельно или раз в две недели – это большая работа. Вы разрабатываете версии целого набора общих компонентов, чтобы люди в командах конечных продуктов могли работать с ними и адаптировать самую новую версию общих компонентов, что, в свою очередь, позволяет с большей вероятностью принять созданную модификацию за основной компонент. Мы организовали такое взаимодействие. И как только это вошло в обычную работу, стало работать «на автомате», эффект тоже был большим, потому что людям проще стало адаптировать компонент, и повышались шансы на то, что их модификация могла быть принята в «основную» линию разработки.

Кроме того, мы добавили определенную ключевую документацию. В по-настоящему успешных проектах свободного ПО есть три отдельных уровня документации. Они могут иметь разные названия или даже быть по-разному организованы, но три уровня выделяются четко. Первый – это общая информация: о чем этот проект и кто в нем задействован. Второй уровень – каким образом начинать использовать продукт: как устанавливать, каковы требования к системе. Подробно все не расписывается и нередко выглядит так: «Обычно использование данной программы начинается следующим образом». А третий уровень – это информация для тех, кто хочет принять участие: как самостоятельно создавать новые версии ПО, каковы программные требования и каким образом вы можете что-то менять или добавлять. Заметьте, что это соответствует трем отдельным группам участников работы: пользователи, помощники и исполнители. Первый, ознакомительный уровень помогает людям решить, хотят ли они стать пользователями, информация по установке помогает им стать пользователями, а третий уровень информации позволяет пользователям стать участниками.

Это применяется в успешных проектах ПО с открытым исходным кодом, и мы скопировали эту систему для предприятия. И вновь эффект был значительным. Например, когда я в первый раз пришел работать с командами, я сел за стандартный компьютер разработчика и попытался начать разработку, но ничего не вышло. Значит, разработчик в команде конечных продуктов, наверняка, тоже не смог бы работать на нем – определенно не хватало какой-то информации. Мне пришлось обратиться к трем разным разработчикам, чтобы он заработал. Мы составили небольшой документ, где четко прописали требования к системе для пользования и разработки и кратко изложили принципы разработки и работы продукта. Эта история опять же возвращает нас к тому, как сделать ПО *простым* в использовании и адаптировать его под свои нужды.

Дженни: **То есть большая открытость в компании изменила стиль работы команд. Но повлияла ли она каким-то образом на то, как они проектируют ПО?**

Ауке: Есть фундаментальная разница между свободным ПО и предприятиями. По-моему, в проектах свободного ПО люди успешно научились повторно использовать ПО, тогда как компании все еще учатся это делать. Компании пытаются сразу проектировать ПО таким образом, чтобы его можно было использовать повторно. В теории это очень здорово, но чаще всего невозможно предугадать, какое именно ПО будет использоваться повторно. Единственное, что точно известно, когда начинаешь разработку программного продукта, это то, что он будет отличаться от того, как ты его себе представляешь, и использоваться будет по-другому и работать совсем не так, как ты планировал первоначально.

Эндрю: **То есть проектирование с учетом возможности повторного использования не всегда благо?**

Ауке: Это не самый лучший подход к разработке ПО. Обычно в проектах ПО с открытым исходным кодом вы сначала разрабатываете что-то для единичного использования. Потом кто-то берет этот продукт и модифицирует под свои нужды. И может повторно его использовать. А когда вы сделали это – использовали и использовали повторно – вам известна унифицированность, и можно заново вывести общую функциональность.

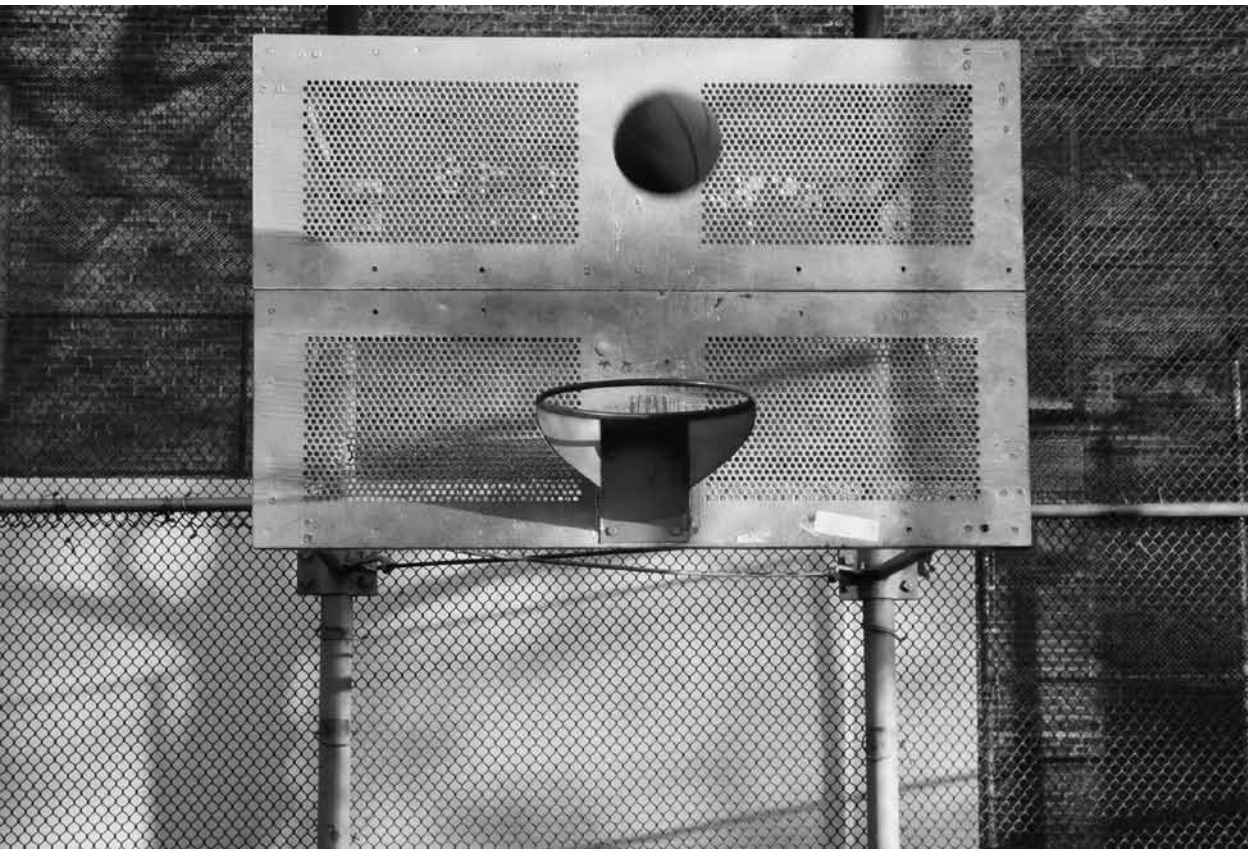
Это происходит довольно часто. Проектирование для повторного использования слишком статично. Оно не работает в таком виде и мало применимо. Возможно, к этому следует стремиться, но вряд ли это достижимо в обозримом будущем. Гораздо лучше схема «использование–использование–повторное использование».

То есть изначально вы проектируете ПО для единственного использования. На предприятиях все происходит так же. В компании сначала разрабатывают определенный алгоритм и характеристику продукта. Затем начинают развивать их, чтобы можно было применить и в других продуктах. Для этого необходим гораздо более открытый процесс, чтобы такая эволюция стала возможной. Этого мы и добились нашим Inner Source.

Эндрю: **То есть проектировать для повторного использования вы не можете? Звучит несколько радикально.**

Ауке: Конечно, можно проектировать и для повторного использования. Я говорю о том, что независимо от назначения проекта цель никогда не бывает статичной, поэтому точное попадание невозможно. В том числе это объясняется и тем, что проектируя и разрабатывая продукт, вы начинаете гораздо больше понимать о нем, а это тоже ведет к изменениям. По сути, ПО с открытым исходным кодом и ПО с исходным кодом, открытым для внутреннего использования, позволяют найти что-то очень близкое к вашим потребностям, хотя и не идеальное, и адаптировать это по необходимости.

В большинстве компаний сейчас принята модель «ПО с закрытым кодом», то есть если вы работаете над каким-то программным продуктом, я ничего об этом не узнаю. Если я попрошу его у вас, вы пришлете мне некоторые исполняемые файлы. Вы можете прислать и исходный код, но пришлете вы только одну версию кода, и, вероятно, это будет версия полугодичной давности. И я не буду ничего с ней делать, потому что если я начну над ней работать, мне нужно будет уходить в полный автоном. Я не смогу направить свою работу вам, потому что версия была настолько старой, что вы не возьметесь внедрять мою модификацию в свое ПО, а получить доступ к новой версии мне будет очень непросто.



II

Цели

Давайте представим, что у вас есть задача и вам нужно ПО для ее выполнения. И давайте представим, что у вас есть два варианта. Первый – блестяще разработанная программа. Она безотказна, у нее прекрасный пользовательский интерфейс и очень хорошая техническая поддержка. Она быстро работает, использовать ее одно удовольствие, она недорога, работает с любой операционной системой и практически на любом компьютере. Но она не может делать то, что вам нужно. Вторая программа ужасна. У нее куча дефектов, она постоянно вылетает и работает крайне медленно. Поставить ее можно только на малоизвестной операционной системе, для которой требуется очень дорогой и быстрый компьютер, в противном случае она не будет работать вообще. У нее отвратительный пользовательский интерфейс, с которым даже простое задание превращается в пытку. Но она может на 20% выполнить вашу задачу. Что вы выберете?

К сожалению, вам придется выбрать плохо написанное ПО, потому что, фактически, оно может делать то, что вам нужно. И хотя выглядит такой расклад довольно смешно, по сути, он не так уж далек от реальности, с которой многие из нас сталкиваются.

Худшие из ошибок в работе команд случаются, когда цели команды начинают расходиться. К сожалению, это происходит гораздо чаще, чем многие осознают. Это не сильно удивляет. Любой, кто листал вузовский учебник по разработке ПО, наверняка видел таблицу, где видно, что стоимость устранения дефекта экспоненциально зависит от срока его обнаружения. Опытным программистам учебник не нужен – они почти наверняка знают это по себе.

Один из самых распространенных способов для проектов разработки ПО попасть в трудную ситуацию – это работа над решением не той задачи. За много лет мы пообщались с десятками разработчиков, тестировщиков, архитекторов, менеджеров проектов и других специалистов, и все они признались, что им знакомо это ужасное чувство, когда вы выдаете заказчику казалось бы окончательный продукт и слышите: «Симпатично, конечно, но разве эта программа не должна была... ?» Всем нам знакомо это чувство, когда все внутри сжимается, оттого что мы узнаем об этой безумно важной функции, о которой прежде никто не говорил. Есть ли среди нас хоть один опытный программист, который не подумал бы: «Если бы мне хоть кто-нибудь сказал два месяца назад о том, что программа должна делать вот *это*, я бы разрабатывал ее совершенно по-другому!»

Если вы хотя бы немного работали в Agile-проекте, то наверняка узнаете фразу «embrace change» (принимай изменение). Один из секретов успешной работы программистов в Agile-командах заключается в том, что они постоянно пересматривают эти цели. Они делают эти цели из-

вестными всем: команда пишет их на доске и специально устраивает встречи, чтобы быть в курсе любых изменений. Они также задействуют заказчиков в ежедневной работе по проекту, потому что это наиболее эффективный способ добиться того, чтобы у всех было единое понимание целей.

Но даже зная о том, насколько такие изменения могут повредить проекту, нам, разработчикам, слишком просто отмести на начальном этапе проекта идею о должной постановке цели. А собственным нашим заказчикам, пользователям и заинтересованным лицам еще проще направить нас по ложному пути, прежде чем у нас вообще появится возможность сказать что-то по этому поводу.

Заказчики – люди, для которых мы разрабатываем ПО, – плохо умеют объяснять, что именно им нужно. Они сразу просят конкретных решений, вместо того чтобы рассказать вам, в чем состоит задача. Они будут просить у вас смирную, резвую и чистенькую лошадь с телегой, когда на самом деле им нужен мощный автомобиль, чтобы добраться из пункта А в пункт Б. У них недостаточно знаний, чтобы попросить машину, преставить улучшенную лошадь и телегу им гораздо проще. Чтобы выбрать лучшее решение, разработчик должен знать общую ситуацию с транспортом.

С одной стороны, у нас, разработчиков, есть свои специфические проблемы. Нам очень просто *думать,* что мы точно знаем, какое ПО мы собираемся разрабатывать. Мы привычно перебиваем заказчика на полуслове, говоря: «Все, я понял, что вам нужно». Затем расходимся по своим рабочим местам и делаем то, что, как нам казалось, требуется клиенту, чтобы затем выяснить, что мы вообще не поняли суть задачи. (Иногда такое случалось и с нами, обычно это происходило, когда мы очень хотели задействовать конкретную крутую новую технологию и, по сути, искали повод применить ее где-нибудь.)

Итак, критически важно понимать цель проекта, а истории и интервью в этом разделе показывают, каким образом это может повлиять на работу наших команд. Одна из самых больших и сложных задач при работе с командами – добиться того, чтобы у всех была единая цель и разрабатывалось именно то ПО, какое нужно. Даже в лучших командах возможны конфликты по поводу целей, и эти конфликты могут развалить любую команду. Но если с самого начала настраивать всех на единую задачу и держать их в курсе изменений в целях – а они *всегда* меняются, – у проекта будет гораздо больше шансов на успех.

9

Создание командного духа

Интервью с Гради Бучем

Немного найдется в мире ПО людей, столь же широко признанных за их заслуги, как Гради Буч. Его работа с Rational и IBM стала направляющей для всей отрасли как в развитии объектно-ориентированных языков и технологий, так и в совершенствовании Унифицированного языка моделирования (UML) и Унифицированного процесса разработки Rational (RUP). В последнее время он много работал с «виртуальными», распределенными командами и распределенными платформами. Об этой работе мы и решили его расспросить.

Эндрю: **Расскажите немного о своей работе, благодаря которой вы изучили механизмы работы в команде.**

Гради: Ого. Хороший вопрос. Знаете, с самого начала своей профессиональной карьеры я был так или иначе задействован в командах с распределенной структурой. Самый первый мой проект начался, когда я еще служил в ВВС. Первые задания я получил на авиавоздушной базе Вандерберг[1], где служил сначала инженером в проекте по разворачиванию телеметрической системы, а потом стал менеджером проекта системы безопасности на полигоне. В обоих случаях мы имели дело фактически с распределенной структурой. Было это в конце 70-х – начале 80-х, то есть это был передовой опыт задолго до начала коммерческого применения таких структур; по сути, мы работали над объединением данных с радаров, разнесенных по всему миру. Уже тогда у меня появился некоторый опыт работы с архитектурой распределенных систем. Кроме того, ребята, работавшие над созданием этих систем, большей частью находились на территории базы Вандерберг, но были и удаленные группы, разбросанные по разным точкам, поскольку сама природа этой системы предполагала такую распределенность. На проекте системы мониторинга безопасности на полигоне, которым я руководил, нам нужно было координировать свою работу с группами в центре Кеннеди[2], то есть мы находились в разных концах страны. Получается, что с самого начала я имел дело с разработкой, ведущейся в разных географических зонах и часовых поясах.

Потом, когда появился Rational, я в большей степени занимался вопросами архитектуры. Я в основном помогал заказчикам переходить от каскадных (waterfall) методов разработки к итеративным (agile), и от монолитных архитектур к более гибким вариантам. Значительные изменения в работе своих заказчиков я увидел только лет 10–15 назад. Работа над многими проектами не могла далее продолжаться в том же офисе или в той же компании. Я видел, как команды распадались по экономическим причинам, если аутсорсинг или ниаршоринг[3] был предпочтительнее. Кроме того, начали разрабатывать вспомогательные системы, а это значило, что не просто одна команда занималась конкретным элементом системы, но необходимо было работать с командами по всей стране, а то и миру. В это время я начал работать еще и с междуна-

---

[1] Военная база США с космодромом, расположенная в Калифорнии. – *Прим. перев.*

[2] Космический центр имени Джона Ф. Кеннеди на острове Мерритт (Флорида, США). – *Прим. перев.*

[3] Nearshoring – аутсорсинг IT-задач в страны, расположенные географически близко к стране заказчика. – *Прим. перев.*

родными корпорациями, где приходилось иметь дело с системами, в работе над которыми были задействованы и разные компании. То есть естественное развитие технологий – самого типа создаваемых систем – и экономики в целом привело организации к необходимости распределения ресурсов как во времени, так и в пространстве. Собственно этот опыт и сформировал мои взгляды.

Эндрю: **Что самое сложное в работе распределенной команды: разное местоположение, разные характеры участников или разница в технологиях?**

Гради: Интересно, что вы об этом упомянули. Да, некоторые технические аспекты до сих пор очень сложно решать, например проблемы с координацией работы в условиях разных часовых поясов. Но, по-моему, настоящие сложности возникают исключительно в социальных аспектах. Я бы разбил их на две категории: во-первых, это вопрос доверия, а во-вторых, культурные различия.

По поводу доверия – если у меня есть географически распределенная группа сотрудников и я не имею возможности работать с ними лично, очень трудно оценить их способности и понять, как они будут реагировать на определенные вещи и как себя лучше поставить с ними. Это очень отличается от той ситуации, когда вы работаете с человеком в одной комнате и видите его каждый день. Так называемая проблема «курилки» – в ситуации с географически разнесенной командой у вас нет шансов на случайные пересечения и дополнительные контакты, а именно такое, свободное и случайное взаимодействие играет большую роль в создании доверительных отношений.

С точки зрения культурных различий, когда я имею дело с системой, работающей на разных концах страны, культура разработки на Западном побережье несколько отличается от культуры восточных штатов, не говоря уже об огромных различиях с культурами Индии, Китая или Японии, и постоянно всплывает множество нюансов, которые могут усложнять процесс разработки.

По-моему, именно эти два вопроса, вопрос доверия и вопрос культурной разницы, наиболее сложны для решения.

Дженни: **Как проявляется отсутствие такого доверия в распределенной команде, в отличие от коллективов, работающих вместе?**

Гради: Знаете, когда члены команды друг другу не доверяют, эффект примерно один и тот же, работают они вместе или удаленно друг от друга. Если они работают рядом и плохо друг к другу относятся, может начаться крайне неприятная борьба, потому что они взаимодействуют напрямую. В распределенных командах это скорее пассивная агрессия,

когда вы как бы игнорируете другую группу и делаете все по-своему. В итоге некоторые группы занимают полярные позиции и лишают друг друга права голоса.

Когда люди не доверяют друг другу, это очень четко чувствуется. В командах, работающих вместе, я порой физически ощущал это напряжение и агрессию. Это сильно сказывается на качестве работы. Недавно вышла замечательная статья, интервью с Энди Херцфельдом о его работе в Mac. Он заметил, что стандартные методы ведут к созданию стандартных продуктов. Это не всегда плохо, есть очень много типовых вещей, которые тоже нужно создавать, но когда вы оказываетесь в ситуации, где необходимы прорыв и новаторство, а в коллективе нет общей культуры доверия, то никаких инноваций не получится, это точно.

Каким образом можно создать доверительные отношения? Есть простые идеи, и есть идеи нетривиальные. Простой способ таков: нужно иногда собирать людей вместе. Придумать, каким образом можно собрать вместе распределенную команду, хотя бы раз, или делать это регулярно, перемещая людей, – это очень действенно. Есть более необычные пути: в IBM провели очень интересную работу в виртуальных мирах, где люди создавали программы, рассчитанные на работу во внешнем мире, и команды, хотя и разнесенные географически, работали вместе над общей задачей, которая не имела отношения к разработке ПО, но таким образом у них была возможность лучше познакомиться.

Такие вещи могут создать фундамент для доверия. Но это не решает по-настоящему проблему «курилки», потому что для ее решения нужны такие условия, в которых люди могут случайно сталкиваться, пересекаться. Поэтому при проведении видео- и телеконференций совсем неплохо, если люди могут поболтать до или после начала: «Как дела, как твой малыш и как поживает твоя собака?» Таким образом, процесс становится «человечным», а разработка ПО, в конце концов, это опыт и работа именно людей. И когда вы можете создать такую атмосферу «человечности», доверие возникнет само собой.

Эндрю: **А как насчет разработчиков, которые хотят, чтобы их просто не трогали, дав им спокойно кодировать в одиночестве? Похоже, с разными людьми к вопросу создания доверительных отношений нужно подходить совершенно по-разному.**

Гради: Несомненно. Есть и такие рыцари кода, которые уходят в работу с головой, не видя ничего вокруг. Но к ним следует относиться очень внимательно: нужно их направлять, мотивировать и постоянно поддерживать интерес в том, что они делают. По сути, лучшее, что можно сделать, – просто не мешать им. Но нельзя отпускать их в полную автономку, потому что они разработают несомненно великие, но совер-

шенно неприменимые вещи. Поэтому необходим тонкий баланс между уважением к индивидуальному стилю работы и интересами дела. Но об этих личностных отличиях знает любой хороший руководитель.

Эндрю: **А что если вы сами разработчик в команде и не знаете, как обращаться с такими людьми?**

Гради: В таких случаях ищут компромисс. Если мой коллега – вот такой оторванный от жизни, с головой погруженный в мир кода человек, возможно, даже асоциальный, – приходится находить общие точки, где мы сможем взаимодействовать. Это может быть непонятный код, с которым я попрошу его помочь мне. Как разработчик в таких ситуациях я стараюсь заставить его оторваться от кода и посмотреть на свою работу с точки зрения реального мира. Хорошо, когда в команде есть такие люди, но чтобы они по-настоящему заблистали в вашей команде, их необходимо все время подпитывать. А это помогает и в создании доверительных отношений.

Эндрю: **То есть мы говорим в основном об «основах обращения и ухода за командой разработчиков».**

Гради: Иногда, ей-богу, я жалею, что у меня нет диплома гуманитария. Это больше социология, чем техника; не то чтобы технических проблем не возникало вообще, но когда численность команд возрастает, на передний план все больше выдвигаются социальные вопросы.

Эндрю: **Однажды нам с Дженни пришлось из команд по 3–5 человек формировать большие группы по 10, 20, 30, 50, 80 человек. А вы как с этим справлялись? Всегда удавалось?**

Гради: Далеко не всегда. Не скажу в процентах, но есть команды, которые ну никак не стыкуются – и такое бывает не только в ИТ, но в любой отрасли. Одни команды могут слиться воедино, другие нет. Самое важное – обеспечить механизм, остальное они сделают сами. Если слишком жестко диктовать им правила, у людей не остается возможностей для самоорганизации. Но если совсем не натягивать поводья, вы теряете организованность и получаете полную непредсказуемость. Надо понимать, что есть люди, для которых очень важна структурированность. Именно это привлекает людей в крупные компании и отпугивает их от компаний начинающих. При этом склад ума, необходимый для работы в начинающей компании, где вас не связывают никакие исторически заложенные механизмы, радикально отличается от склада ума, требуемого для поддержания этих механизмов. Нередко тот, кто заложил основы чего-то, очень быстро начинает скучать, ему совсем не интересно дальше долгими годами заниматься поддержкой разработанных программ, однако в бизнесе требуется не только это.

Здесь возникает еще одна проблема, я называю это «родовой памятью». У каждой системы есть архитектура: большей частью она бывает случайной, и только некоторые из них создавались намеренно. Я имею в виду, что архитектура в целом строится из десятков тысяч маленьких ежедневных решений. И однажды вы просыпаетесь и понимаете: вот, что у меня получилось. Сейчас уже и Google находится в такой ситуации. Они говорят: «Когда мы основывали компанию, не было никакой преемственности, а теперь мы пришли к этой чудесной системе и лидируем на рынке»; они принимали определенные архитектурные решения для своих продуктов, иные, чем в Amazon и IBM, однако они инвестировали именно в эти технологии и получили то, что у них есть сейчас. И теперь, поскольку мир не стоит на месте, встает интересный вопрос – та архитектура не была подробно отражена в документации, зачастую никакой документации вообще не было, и хотя код верен, он не может быть истиной в последней инстанции, потому что не отражает ни логического обоснования, ни аспектов, по которым принималось компромиссное решение, ни любых других вещей, которые невозможно вывести из отдельных строк кода; есть схемы, выходящие за пределы самих этих строк. Такие вещи обычно сохраняются в «родовой памяти». И в любой организации есть патриархи, которые работают там уже давно и лично помнят эти обоснования и возникавшие междисциплинарные вопросы. Сложность здесь в том, что организация продолжает расти, люди перемещаются, и этот адресат может, фактически, уже находиться в другом месте.

То есть помимо проблемы временного и географического распределения есть проблема утечки интеллектуальной собственности, потому что хранить ее в «родовой памяти» – страшно дорого, точнее, память эта ничего не стоит, но извлечь из нее информацию может оказаться очень затратным, и чем меньше этой памяти, тем она дороже.

Эндрю: **Реально ли извлечь эти знания? Что если привлечь этих, несомненно, очень занятых людей к написанию обоснования какой-то программы?**

Гради: Конечно. У меня была такая история, когда я работал в государственной компании на Восточном побережье. Мне сказали: у нас тут сложности с демографией. Есть совсем юные ребята, только после колледжа, и есть старички. Средней прослойки нет совсем – всех завлек Манхэттен, и они теперь где-то на Уолл-стрит. Теперь старые сотрудники уходят на пенсию, умирают – и не всегда именно в этом порядке, – и мне дали задание записать их рассказы. Взять видеокамеру. Задать им вопросы и выяснить, в каких шкафах какие скелеты. Даже такие простые меры могут выручить.

Дженни: **Вернемся опять к тому, что вы говорили об обосновании для кода и о том, как принимаются решения в командах. Что вы думаете по поводу критики в коллективах, и как этот вопрос решается в распределенных командах? Кроме того, очень важно, чтобы люди понимали, за какую часть работы они отвечают, и чтобы их работа была прозрачной для остальных. Как это решается в распределенных командах?**

Гради: Очень хороший вопрос. Думаю, именно в таких случаях должен применяться механизм, при котором команда далее действует самостоятельно. Мне очень понравилась формулировка, и я перефразирую это так: идеальные условия разработки – это условия, позволяющие разработке течь максимально гладко, а это значит, что вы можете собирать информацию с минимальным вовлечением самих разработчиков, если я могу создать системы управления изменениями, которые будут работать за меня, если я могу сделать так, чтобы при тестировании создавались ежедневные новые версии программы, и так далее. Такие вещи необходимо показывать в разделе «инструменты», потому что они позволяют автоматизировать процесс настолько, что не будет необходимости дергать отдельных разработчиков. И все-таки, вы правы, будет возникать напряженность и будут моменты, когда кому-то придется принимать очень сложное решение. В команде, где нет явного лидера, эта процедура может стать весьма болезненной, потому что у каждого будет свое мнение на этот счет и борьба будет жесткой. Но в итоге в большинстве крепких команд обнаруживаются определенные центры тяжести и оплоты мудрости. Они могут быть самопровозглашенными на основании их опыта. Хороший пример – Линус Торвальдс и весь проект Linux. Он выдвинулся как интеллектуальный лидер, потому что стоял у истоков, но Линус там не один такой. Есть люди, от которых зависят части общего целого, и они благодаря своему опыту получили это право и заслужили уважение. То же самое происходит в любой слаженной команде.

Когда команда расширяется, возникает необходимость прописывать такие роли официально. Поэтому вы видите людей, получающих должность «главный дизайнер» или «архитектор», а в больших системах есть главные архитекторы по целой группе или даже компании. Здесь мы коснулись все более актуального вопроса – где будет размещаться центр контроля при снижении темпов разработки и переходе от ситуации, когда создается невероятное количество нового, но приходится выбрасывать много кода, к ситуации, когда код начинает накапливаться, становится капиталовложением и частью общей культуры и его нельзя больше так просто отправлять в утиль. Некоторые из таких процессов принятия решений приходится прописывать официально.

Дженни: **Вы говорили о членах команды, а как насчет команды в целом? Как вы охарактеризуете слаженную команду?**

Гради: Назову две характеристики, самые важные для меня. Такие вещи сразу чувствуются, как только попадаешь в группу. Главная – эти люди получают удовольствие. Если им нравится работать вместе и нравится то, что они делают, это видно сразу же. И это четкий показатель слаженной команды.

Вторая – они очень чутко реагируют на изменения и при этом могут писать программы, отвечающие деловым потребностям. Это сложное равновесие, о котором я уже говорил. Есть небольшие команды, работающие без всякого контроля и ограничений, получающие огромное удовольствие, прожигая чужие деньги и не делая ничего полезного, но команда, понимающая, что требуется для дела, наладившая хорошее взаимодействие и отлично умеющая программировать, – это совсем другое. Они будут разрабатывать именно то, что нужно для бизнеса.

Мне кажется, это две самых важных вещи. Все остальное менее значимо, если честно. Есть куча разных подробно расписанных параметров, но эти два стоят выше остальных.

Эндрю: **Это непростая задача. Наверняка многие работали в командах, где им очень нравилось, но немногие могут похвастаться тем, что эти команды были по-настоящему продуктивными.**

Дженни: **Да, обычно эти вещи исключают друг друга.**

Эндрю: **Да, как этого избежать?**

Гради: Магию трудно создать. Она самозарождается и так же легко уходит. В свое время Xerox PARC была таким магическим местом, хотя они почему-то так и не смогли начать на этом зарабатывать. Мне кажется, в Apple сейчас именно такая ситуация, когда разработчики отлично проводят время и при этом они очень хорошо заявлены на рынке. Google в свою раннюю пору был местом, где большой акцент делался на хорошую атмосферу, при этом они зарабатывали нереальные деньги, и никто не замечал неэффективности в работе. Теперь задачи усложняются. IBM – компания, где бизнес приравнивается к удовольствию, однако там немало возможностей и для развлечений, позволяющих людям сплотиться и полюбить то, чем они занимаются.

То есть я не знаю, как к этому приходить. Вы спрашиваете, каковы правила, по которым можно это создавать. Если бы у меня был ответ, я бы давно уже стал владельцем маленькой страны.

Эндрю: **То есть надо говорить не о правилах, а о каких-то вещах, которые могут подтолкнуть вашу команду в правильном направлении или задать общий дух?**

Гради: Точно. И две из таких вещей я назову.

Первое – давать им возможность отрываться от работы. Иными словами, есть группы, с головой погруженные в процесс программирования, и это чудесно, потому что вам, конечно же, нужны люди с техническими навыками. Но, в конце концов, мы пишем программы потому, что они кому-то нужны, а самим нам нужно, чтобы команда не была полностью изолированной, оторванной от мира. Мне кажется, это важно, потому что дает сотрудникам общий контекст и понимание логики того, что они делают.

Второе, что нужно делать, особенно в качестве менеджера проекта, – защищать их от остального мира. Многие компании задействованы в разных не самых приятных политических делах. Но если команда будет думать о ежедневных махинациях на политической арене, это напугает их до смерти. Поэтому хороший проектный менеджер должен стать щитом между разработчиками и политикой компании.

В некоторых случаях, как я уже говорил, лучшее, что может сделать менеджер, – это убрать всех и вся с пути этих людей. Я говорю о простых вещах вроде этих идиотских совещаний, необходимых при росте компании, – но ведь, с другой стороны, вам не нужна и полностью неконтролируемая команда. То есть, конечно, здорово, когда люди рубятся в коридорах игрушечными саблями, но хорошо бы они не слишком отвлекались от работы.

Дженни: **Как руководитель может обеспечить концентрацию своей команды?**

Гради: На ум приходят три вещи. Первое – это то, что лучшие из виденных мною в этой отрасли руководителей очень хорошо владеют речью. Они отлично умеют писать код, но могут и говорить с людьми на другие темы – могут подняться выше самого процесса программирования.

Второе – это то, что для них важны личные взаимоотношения. Лучший лидер – не просто технический гений, он хорошо понимает, что разработка ПО – это человеческий труд, и уважает этих людей. У людей бывают выходные – у них умирают собаки, дети приносят из школы двойки – и все это влияет и на команду, и на разработку. Хороший руководитель понимает такие простые человеческие факторы.

Третье – хороший руководитель может действовать одновременно на нескольких уровнях абстрагирования; люди, способные разобраться в коде, а потом пойти и пообщаться с директором, встречаются очень

редко, но это самые лучшие руководители, потому что они умеют правильно изложить ситуацию.

Эндрю: **Любопытно. Не раз в юности, работая в коллективе, я думал: кажется, мы говорим с этими начальниками на абсолютно разных языках. Мне определенно был нужен «переводчик», кажется, я даже стал менеджером из чистого упрямства и злости, не желая ударять в грязь лицом. Что бы вы посоветовали людям, которые оказались в такой роли просто потому, что некому кроме них было взять ее на себя?**

Гради: Я всегда даю таким людям один совет: пожалуйста, играйте эту роль мягко и без высокомерия. Вы имеете дело с очень продвинутыми технически людьми, у каждого есть чувство собственного достоинства, и больше всего людей раздражают заносчивость и отсутствие уважения. Поэтому берите на себя эту роль спокойно и мягко, но смело; не боясь неудачи.

Есть еще одна вещь, которую вы должны осознавать на этой позиции, и я понимаю, что для начинающего это непросто, потому что есть опасность потерять работу, но очень важно иметь возможность говорить правду вышестоящим. Говорить руководству: «Это безумный график, ни один человек его не выполнит. Это неосуществимо». Нужно научиться честности. Иначе вся ваша работа превратится в снежный ком вранья. Важно говорить начальству правду. Это еще один фактор в защите вашей команды от всей этой странной политики.

Но идти на такое противостояние непросто; иногда руководство вообще ничего не понимает в разработке ПО. Для них программы – просто еще одна крупная статья расходов компании, и они предпочли бы отдать ее во внешний подряд. В таком случае хороший менеджер должен стать человеком, мягко проводящим ликбез для высшего руководства, а это процесс нескорый. Но если не получится, всегда можно найти другую работу: жизнь слишком коротка, чтобы работать на плохие компании.

Эндрю: **Надо же, мы с Дженни очень много беседовали с людьми в разных компаниях на тему того, почему проекты бывают неуспешными. Фактически, у нас получился сборник «Почему проваливаются проекты», который можно издавать. И мы постоянно сталкивались с тем, что человек начинает неловко себя чувствовать, если с ним заговорить о провалах. А зря, потому что если хочешь извлечь из неудачи урок, нужно говорить о ней. Вы замечали такое?**

Гради: Один из признаков, по которым я оцениваю общее состояние компании, – замалчивание провалов. Самые занудные и панически боящиеся неудачи организации обычно наименее изобретательны, и работать там не в радость, потому что они настолько боятся провала, что выбира-

ют самые консервативные пути. С другой стороны, компании, признающие возможность неудачи, то есть, конечно, не полного краха компании, но просто более свободно относящиеся к промахам, более продуктивны, потому что они не трясутся за свою жизнь над каждой строкой кода. Потому что, вы же понимаете, в любую строку может вкрасться ошибка. Нужно иметь возможность со временем ее исправить.

Это большая проблема в организациях, переходящих от разработки чего-то нового к историческому этапу поддержания созданного. Я видел повторение этого сценария много раз. Есть группа, особенно это характерно для команд, использующих что-то легкое, вроде Ruby on Rails, и никакого технического наследия прошлого у них нет. Ну, может, есть где-то база данных, но кто-то ставит задачу «разработать X», и они разрабатывают X, потому что больше им работать не с чем. Возможно, есть несколько сервисов или программных интерфейсов, но большей частью они движутся по девственным лесам. Проблема в том, что создаваемое начинает накапливаться. Каждая написанная сегодня строка кода назавтра становится «исторически существующей». И вы приходите к точке, когда проще сказать: «У, выбросить все и начать заново», – но потом вы приходите и к точке, где приходится признать: «Я уже не могу себе позволить все это выкинуть, слишком многое сделано. Переписать все заново уже не получится». И вы подходите к моменту, когда нужно делать выбор. Можно полностью отказаться от разработанного и начать заново, но это не всегда целесообразно. Другой вариант – упаковать и убрать в дальний ящик, мол, пусть полежит, оно неплохо работает, однажды к нему вернемся. Хотя на самом деле не возвращаешься никогда. И есть случаи, когда вы хотите извлечь максимум: отказаться, но взять из него лучшее, переписать, но не с чистого листа.

Хорошо держащиеся на плаву организации, по моему опыту, постоянно перерабатывают, делают рефакторинг (реорганизацию) разработанных программ, и не только на уровне кода, но и на уровне архитектуры. Но тут всегда есть та опасность, что вышестоящая организация скажет: «Мы уже оплатили этот код, почему мы его переписываем?» И ответом будет: потому что хотим упростить его, что даст нам возможность новаторства и степень свободы, которые в противном случае недостижимы. Но убедить людей в этом непросто, и это входит в «воспитательный» процесс, о котором я уже говорил.

Дженни: **У кого-то слово «рефакторинг» автоматически вызовет ассоциации с итеративной Agile-разработкой. Вы об этом говорите? Как это влияет на производительность команды?**

Гради: Для меня это признак лучших методик, которые я наблюдал за долгое время. Как бы вы ни назвали это, лучшая практика, кото-

рой следуют самые гиперпродуктивные организации в этой области, заключается в следующих трех аспектах. Эти организации пошагово и итеративно создают версии исполняемых архитектур. Аспекты же такие.

Во-первых, акцент на исполняемости. Выдавать сырой код в качестве основного продукта очень важно и, несомненно, это имеет отношение к Agile. Во-вторых, то, что это делается пошагово и с итерациями, означает, что у вас есть заданный ритм, куда вы можете встроить рефакторинг, и в этом случае вы можете себе позволить некоторые неудачи в процессе разработки. Кроме того, у вас есть параллельные пути, чтобы пробовать новое. Третий элемент, который меньше относится к семейству Agile, а в большей степени к RUP, это внимание к архитектуре как способу управления. Потому что я по опыту могу сказать: как только начинаешь иметь дело с системами любого значительного уровня сложности или масштаба, появляется набор накапливающихся проектных решений, которые попадают в родовую память, и для меня слишком дорого, если вообще возможно, выбрасывать эти решения и начинать все заново. И я с ростом системы пытаюсь управлять процессом сохранения этих решений по архитектуре, чтобы в итоге на руках у меня оказалась не полностью случайная программа, но продуманная в течение долгого времени.

Вы можете называть это Agile или как вам угодно, но именно такие практики последовательно применяются в действительно производительных и эффективных командах.

10

Если вы провалились

Дженнифер Грин

В моем резюме указано: «Дженни Грин – опытный менеджер по проектам и развитию, осуществившая ряд успешных проектов». И это правда. Я управляла многими командами, которые сумели создать нужные программы требуемого качества, уложившись в сроки и смету. Думаю, многие могут похвастаться тем же. Но когда я вспоминаю все, чем занималась около пятнадцати лет, то понимаю, что многие из этих успешных проектов не улучшиши положение на рынке компаний, на которые я работала. Иногда они даже не решали те проблемы, которые были призваны устранить. И я прочитала достаточно, чтобы понять, что и в этом не одинока. Многие проекты приводили к еще худшим результатам, чем мои. Некоторые опубликованные оценки свидетельствуют о том, что возможность провала программного проекта составляет около 80%.

Несколько лет назад мы с Эндрю начали посещать конференции и другие сетевые мероприятия с лекциями на тему «Почему проваливаются проекты». Нашей целью было помочь установить связь между тем, что считается трудным или сложным (проверка кода, оценка), и тем, как избежать обычных ошибок. Мы считали, что лекции должны проходить в живой, неформальной обстановке, потому что собирались использовать для иллюстраций кое-что усвоенное нами при написании книг «Head First». И мы получили то, что ожидали, попутно задев за живое тех, кто, казалось, умирает от желания побеседовать об ошибках в своих проектах. То есть многие хотели знать: «Нет, правда, *почему* мой проект провалился?» Чаще всего беседы приводили к тому, что нам начинали изливать душу: рассказывали истории о безумных решениях руководства, о жутком программисте, который из вредности не испра-

вил свой код, о продукте, над которым с самого начала не надо было работать. В присутствии слушателей мы всегда старались направить разговор к конкретному примеру, который мог помочь («Если у вас проблемы с качеством, то проверяйте код заранее» или «Все время не успеваете к дедлайну? Используйте консенсусные методы оценки – например Wideband Delphi»). Эти советы действительно очень помогли нам с Эндрю в смысле карьеры. Но они не могут решить межличностные проблемы или понять, насколько тот проект, над которым вы работаете, подходит вашей компании. Они не гарантируют успех. И наедине мы часто говорили об этом.

Я работала с компаниями, которым так нужно было постоянно производить какие-то программы, что они забывали о том, зачем это, собственно, нужно. Там, где тратят кучу денег на создание программ, просто чтобы доказать, что им есть над чем работать. Я видела проекты, начатые без четкого понимания того, что же, собственно, будет производиться, и встречалась с маловменяемыми руководителями, которые выжимали из команды все соки, чтобы успеть к нереальному дедлайну и закончить не столь важный на самом деле проект. Я работала по ночам, по выходным, отказывалась от отпуска из-за программ, которые потом почти не использовались. Я знаю, что такое работать в команде, цели которой постоянно меняются и нет гарантии, что то, что вы делаете, вообще кому-то понадобится. Можно ли считать успешным проект, если вы вовремя сдали бесполезный продукт? Если ваша команда не разбежалась на условиях, подобных вышеупомянутым? Честно говоря, трудно быть действительно успешным в такой ситуации. Лучше всего попытаться согласовать нужды и сроки. Боритесь за то, чтобы предоставить сотрудникам более разумные цели, хотя иногда в этой борьбе нельзя победить.

Многие из лучших команд, в которых я работала, были поставлены в невозможные ситуации, так что самая замечательная работа объективно не приводила к успеху. Но люди оставались со мной, и иногда в самых невозможных ситуациях рождались новаторские решения. Переехав в Нью-Йорк в самом начале карьеры, я попала в команду, запомнившуюся мне как коллектив самых трудолюбивых и прекрасных людей, с которыми я когда-либо работала.

«Мы в Gabfest Software считаем, что если посадить несколько очень умных людей в одну комнату и задать им очень тяжелую задачку, можно получить потрясающий результат».

Помню, с каким вдохновенным лицом произносил эти слова технический директор. Он был полон энтузиазма по поводу того, чем они занимались, и это просто обезоруживало. Я проходила собеседование на должность менеджера по качеству в небольшом стартапе, который про-

изводил удивительные вещи. Они собирались изменить способ общения человека с компьютером: научить компьютер воспринимать человеческую речь. Эта неформальная беседа велась на исходе дня, после того как я встретилась с половиной компании и ответила на более традиционные для собеседования вопросы. Он пригласил меня в кофейню рядом с офисом и стал излагать свой взгляд на компанию и ее будущее. Когда он произнес слова об умных людях и тяжелых задачках, они показались мне не просто разумными, но даже вдохновляющими. Умные могут всё, правда же? В то время я считала себя очень умной. Протестировав пару лет внутреннее ПО в очень консервативной финансовой компании, я почувствовала, что способна на большее. Я ушла из той компании и приготовилась к большому бою.

Однако от кодирования естественного языка мой трудовой опыт был крайне далек. Компания Gridline, из которой я только что пришла, была небольшой фирмой, которая делала финансовые программы для отслеживания акций. Я поступила туда после переезда в Нью-Йорк (до того я несколько лет работала в онлайн-сервисе, тестируя клиентские программы на C++). Работать в Gridline было всем хорошо, но скучно. В одежде там предпочитали не casual, a business casual: все ежедневно носили костюмы и галстуки, являли собой типичных финансистов и, в основном, работали в этой сфере уже давно.

Компания была небольшой. В руководстве преобладали те, кто стоял у истоков. Они гордились успехами компании и проявляли такую лояльность к ее основателю, какой я больше нигде не встречала. Многие сотрудники по нескольку лет занимали одну и ту же должность. Я работала там менеджером по качеству, тестируя их программы по очистке данных. Пару лет мне было там вполне комфортно. Я подобрала и обучила с нуля команду тестеров, спроектировала и создала первые версии процесса разработки. Это сильно повлияло на качество их работы. Когда команда тестеров была создана, я стала брать на себя все большую ответственность. Однако это не сопровождалось позитивными изменениями в платежной ведомости. Внезапно до меня дошло, что лучше бы попробовать себя на какой-то другой работе. Бум доткомов был в полном разгаре, и любой, кто знал хоть что-то о компьютерах, мог запросто найти приличную работу.

Поднявшись на лифте в офис Gabfest, я была поражена тем, что эта работа, казалось, была диаметрально противоположна той, с которой я ушла. Не было ни охраны, которая фиксировала время вашего прихода, ни стойки с ключами, ни роскошного вестибюля. Никто не носил костюм и галстук. Весь офис представлял собой одну большую комнату, полную столов. Не было ни отсеков, ни стен – никакой частной жизни. Я вспомнила, как отстаивала свое мнение при переделке офисов

в Gridline. Офисная мебель там отражала ваш статус; маленькие отсеки предназначались для новичков и временных сотрудников, большие – для тех, кто проработал уже несколько лет; маленькие офисы со столами из формайки – для менеджеров и супервайзеров, а большие, с настоящими деревянными столами, – для директоров, вице-президентов и выше. Мы вели нескончаемые диалоги по поводу того, кто где должен сидеть и кому какую офисную мебель дать. Все это значило в Gridline очень много. Те, кто работал в отсеках, серьезно готовились к тому, что однажды им дадут настоящий офис, а начальство действительно гордилось своей деревянной мебелью. Я свое место знала: мебель из формайки и еще лет пять до деревянной. Но в офисе Gabfest никто явно не беспокоился по таким идиотским поводам. Мы все должны были работать вместе. Два года понаблюдав борьбу за офисы, должности и обязанности, я наконец-то поступила на работу, на которой можно было просто работать.

Об издержках доткомовского бума было написано уже многое, и когда я прочитала о других стартапах того времени, то поняла, что в отношении прелестей подобной работы я вытянула пустой номер. Не было у нас ни настольного футбола, ни массажиста для сотрудников, ни повара с натуральным питанием, ни даже бесплатных обедов. Просто большая комната, полная людей, которые очень-очень активно работают над какими-то невероятными задачами.

Они создавали продукт, который я до сих пор до конца не понимаю. Группа компьютерных лингвистов создала устройство по обработке речи: в него вводили вопросы и оно моделировало ответы, основываясь на глубоком понимании обсуждаемого предмета. Первый продукт, который они создали, знал все что только можно о ноутбуках. Если вы вводили фразу «Мне нужен быстрый и недорогой ноутбук весом меньше пяти фунтов», формировался SQL-запрос к базе данных и выводился список ноутбуков, отвечавших этим требованиям. Программа предназначалась для веб-сайтов, которые хотели бы обеспечить более индивидуальный подход к своим клиентам, не нанимая специальных людей для работы с ними.

У истоков стоял один из старших сотрудников нашей компании. Он возглавлял команду лингвистов и построил прототип программы еще до того, как была сформирована компания. Он очень трепетно относился к своей программе и все время пытался ее улучшить. Но он не считал, что это когда-нибудь станет коммерчески жизнеспособным продуктом. Сейчас-то я понимаю, что это должно было послужить тревожным сигналом для всех сотрудников. Но всем нам нравилась наша работа. Я имею в виду, что всем нам нравилось то, чем мы занимались в данное время, так что направление всего проекта и его общая цель выглядели менее важными.

Новые знакомые мне сразу же понравились. Они были умны и забавны, часто впечатляюще образованны и полностью преданны своей работе. Программа в основном работала. Пока диалог держался в рамках ноутбуков, она серьезно помогала при продажах. На качестве оборудования группа сосредоточилась с самого начала. Вместе с продуктом они создали автоматический инструмент для тестирования, чтобы отлавливать ошибки в разработке как можно раньше. Было ощущение, что работающие здесь так поглощены своей задачей, что каждый день прикладывают максимум усилий для расширения возможностей программы.

Сейчас можно сказать, что с программой, которую мы делали, было множество проблем, и не всегда технических. Например, мы хорошо усвоили, как продавать через Интернет ноутбуки, и было совершенно очевидно, что никто не делает покупки так, как хотелось бы Gabfest. Если вы хотите купить по Интернету ноутбук, то можно воспользоваться сведениями о конфигурации, которые предоставляет большинство компаний-продавцов, и выбрать из предоставленного списка. Собственно, такой вариант гораздо удобнее, чем та программа, которую мы создавали. Признаемся себе: далеко не каждый захочет вести диалог о ноутбуке с программой распознавания речи, чтобы определить те шесть или семь опций, по которым можно его выбрать. Но лингвистам эта проблема была интересна и могла в конце концов коммерчески оправдаться.

В этой компании я действительно работала много. Неважно, с чем я имела дело – с демоверсией или готовым бизнес-продуктом, – я до смешного долго копалась, чтобы убедиться, что все работает правильно. Помню, как просиживала ночами, работала все выходные – притом почти постоянно. Как, собственно, и все. На любого сотрудника можно было положиться в такой степени, какой я позже просто не встречала. Мои обязанности заключались в отслеживании поведения продукта, наиболее часто повторяющихся ошибок и планирования работы остальных сотрудников отдела качества, чтобы уложиться в самые жесткие сроки. Поскольку собственно тестирование с самого начала было автоматическим, мне нужно было найти лучший способ оценки всех результатов тестирования и отслеживать их. Я начала отыскивать типичные проблемы в процессе разработки и вместе со своей командой работала над их устранением. Мне это было интересно, и именно здесь я стала применять на практике те советы, которыми впоследствии пользовалась в управлении командами.

Все предложения, которые, на мой взгляд, могли помочь нашим продуктам, я сообщала непосредственно техническому директору, руководителям разработки и самим разработчикам. Я могла предложить изменение и тут же увидеть его результаты. По каждому изменению мы

спорили, прежде чем его применить, но люди тут не страдали зашоренностью, так что всё, что на вид имело смысл, можно было пробовать. Я доверяла тем, с кем работала, а они доверяли мне, и все было прекрасно. После работы в местах с жесткой иерархией было приятно получить доступ к тем, кто принимает решения.

Но несмотря на всю прелесть такой работы, у нее были и серьезные недостатки. Помимо проектов обработки естественного языка, были и проекты искусственного интеллекта, осуществляемые без участия команды лингвистов. У нас была также команда сбора данных, которая должна была записывать диалоги пользователей с программами. Часто вспыхивали ожесточенные споры о том, стоит ли нам преследовать сразу несколько целей.

С ростом компании три эти группы превратились практически во враждующие фракции. Все горячо верили в собственный проект и были искренне убеждены, что именно от их задания зависит победа или поражение компании на рынке. Компьютерные лингвисты сражались за то, чтобы улучшить программу распознавания речи, расширить ее словарь и области знаний. Настоящего смысла в этом не было. Есть неограниченное количество способов выражения идеи, даже если это идея «хочу быстрый ноутбук». Они понимали, что их программа никогда не будет столь же умна, как человек, но понимали также, что машину нельзя считать работающей нормально, если она не понимает, что «хочу, чтобы мой ноутбук был быстрым» и «не хочу, чтобы мой ноутбук был медленным» – одно и то же. Такую задачу легко описать (особенно тем, кто с ней не сталкивался), но решить ее с лингвистических позиций чертовски сложно. А это и пытались сделать компьютерные лингвисты. Они считали, что чем богаче возможности их программы, тем лучше конечный продукт.

Специалисты по искусственному интеллекту хотели, чтобы программа могла расширять словарь и даже усваивать новые типы поведения и идей у людей, которые с ней разговаривали. Их работа была не столь сложна, как у лингвистов: они пытались наложить свои функции на уже работающее средство обработки речи. Но несмотря на более скромные цели, этот проект выигрывал в крутизне. Эта идея всем нравилась: они хотели, чтобы программа становилась умнее не из-за того, что ее дорабатывают, а просто от того, что с ней разговаривают.

Если бы компания действовала только в двух направлениях, в которые ее тянули лингвисты и специалисты по искусственному интеллекту, все могло бы повернуться иначе. Но была и третья фракция, которая преследовала собственные цели: сбор данных. Они хотели, чтобы программа провела диалоги со всеми пользователями, что было бы впоследствии использовано для целевой рекламы, направленной на определенных

клиентов и основанной на текущих диалогах. Все интуитивно чувствовали, что здесь таится мощнейший потенциальный источник дохода. Правда, все мы – даже сами сборщики данных – беспокоились по поводу возможных проблем из-за вторжения в тайну переписки. Но это был один из козырей продукта: клиентов Gabfest привлекала идея больше узнать о своих покупателях, основываясь на их диалогах с программой.

Короче, это была битва трех армий. Лингвисты думали: «Все они не нужны, если у нас не заработает программа». Цель команды интеллекта не была продумана надлежащим образом, но все считали, что она достойна, потому что это так круто. Сборщики данных считали себя основным источником дохода для компании. И при всем нашем уме все эти задачи были очень трудны. Все мы чувствовали, что компания пытается одновременно решить слишком много задач. Но это было частью общей цели, которая должна была привести продукт к успеху. Однако руководство поощряло, чтобы все три фракции двигались в своих направлениях, далеких друг от друга.

Все кончилось тем, что у нас возник ряд почти «конспиративных» проектов, запущенных одновременно. Команда искусственного интеллекта создала множество прототипических нейронных сетей, но мало что добавила к основному продукту. Компьютерные лингвисты добавляли все больше функциональности к существующему продукту, хотя рейтинг его продаж был критическим. И это очень хорошо, на мой взгляд, отражало доткомовский бум: суть его была не в настольном футболе, не в гибком рабочем времени, не в красивых названиях должностей, не в открытых офисных планах, бесплатном питании или даже взлете и падении венчурного капитала. Суть его была в том, что люди могли пойти на больший риск и заняться даже более интересной работой, чем представляли. В это время поощрялся творческий импульс. Для нас это означало настоящий научный поиск в области лингвистики и искусственного интеллекта с целью создания реальной бизнес-модели.

Все продолжали работать над своими задачами. Но с течением времени оказалось, что они никак не укладываются в рамки одного продукта. Полагаю, большинство из нас об этом подозревали, но гнали такие мысли из головы, потому что все мы учились чему-то, и это была самая увлекательная работа в жизни. Можно сказать, что все мы выбрали себе приключение и получили за него награду – отраженную не столько в платежной ведомости, сколько в дальнейшей карьере. Все сотрудники Gabfest гордились тем, что делали. Они отстаивали свои мнения, считая, что их работа обладает высокой ценностью. Каждый пытался как можно больше повлиять на деятельность компании. Часто вспыхивали жаркие схватки в конференц-залах: все орали друг на друга по поводу того, куда движется компания.

Как-то днем во вторник технический директор сказал, что у компании кончились деньги и через две недели она закроется.

Помню, что в тот же вечер, уйдя с работы, мы несколько часов обсуждали, что делать. Мы взяли пива и устроили вечеринку: впервые за долгое время получился настоящий вечер отдыха. Больше не было никаких схваток. У компании больше не было цели. Мы стали людьми, которые вместе прошли через испытания.

Потом случилось нечто странное. В последние две недели работы компании все работали как ни в чем не бывало, как будто ничего не изменилось. Мы просто старались сохранить продукт в максимальном объеме. Ни у кого не возникало даже сомнений по этому поводу. Мы работали по 12 часов в сутки, даже зная, что через две недели останемся без работы. Все так беспокоились о том, чем занимались лично они, что было неважно, что станет со всей компанией. Появилась новая цель: сохранить всю свою работу, так чтобы к ней можно было вернуться позже. И в какой-то мере, думаю, все мы верили, что так оно и будет.

Даже потом мы помогали друг другу найти работу. Мы давали друг другу рекомендации. К моменту закрытия компании одна моя подруга была уже на девятом месяце, и вся компания даже после развала устроила вечеринку в честь рождения ребенка, собрав последние деньги на подарки. Месяцы и годы спустя мы не теряли контакт друг с другом. Несмотря на всю ущербность исходной идеи, мы были так увлечены собственной работой над ней, что, думаю, все с теплотой вспоминают опыт тех дней и наши отношения. Эта компания сплотила нас. Я могла с легкостью дать рекомендацию практически любому из тех, с кем работала в ней. В моей карьере не было другой компании, о которой я могла бы сказать то же самое.

Уверена, что ни в одной другой моей команде люди не были так преданы своей работе. Это по-прежнему меня поражает. Но оказалось, что ума, смекалки и денег недостаточно, чтобы решать большие задачи. Для начала требуется единая жизнеспособная цель. Мы не смогли ни приспособить обработку естественного языка к нуждам рынка, ни создать продукт, который сделал бы процесс онлайн-покупки более личным для каждого. Мы не смогли революционизировать способ общения человека с компьютером. Но наша работа в Gabfest была в своем роде магией.

Несмотря на общую негативную оценку наличия в карьере проваленных проектов, резюме почти всех сотрудников полны предложений, подобных процитированным. В моем случае эта неудача была ценнее многих достигнутых с тех пор успехов. Так можно ли считать ее успехом? Безусловно, лично мне она помогла вырасти. Но с объективной точки зрения компания и ее продукты провалились. Насколько я знаю, к тем

программам так и не вернулись. Да, мы были достаточно умны и квалифицированны, чтобы создать любую программу, которую хотели: команда Gabfest состояла из самых умелых сотрудников, с которыми я когда-либо работала. Но мне встречались гораздо менее квалифицированные специалисты, которые тем не менее сделали кучу денег на программировании, потому что выбрали тот продукт, который может продаваться, и вышли на таких клиентов, которые его покупали.

В общем, никому не стоит считать неудачи, которые потерпел в жизни и на работе, своим недостатком. Я, кстати, вообще не думаю, что это что-то плохое. Так что и эту мою неудачу не записываю в минус.

11

Планирование

Интервью с Майком Коном

Каждый, кто следил за миром программного обеспечения начиная с конца 90-х, по крайней мере слышал об Agile-движении. Agile-разработки связаны с нашей многолетней работой методами, применяемыми в Agile-проектах, – в особенности методами планирования и оценки, а также влиянием этих методов на команды и создаваемые программы. Майк Кон написал несколько книг и часто выступает по этой теме.

Эндрю: **Для начала просим вас рассказать о себе. Как вы научились тому, что знаете сейчас о работе в команде?**

Майк: Наверное, мне сразу повезло – на одной из моих самых первых программистских работ. Мой босс не знал, что с нами делать. Он совсем не умел руководить программистами и в итоге назначил ответственным меня, хотя у меня не было нужного опыта. Но я знал чуть больше остальных коллег – нас в то время было четверо. В итоге я стал чем-то вроде лидера команды. С тех пор я работал во многих командах программистов и побывал вице-президентом по инжинирингу в нескольких компаниях. Поэтому могу порассказать о разных компаниях.

Дженни: **Не могли бы вы обобщить свой опыт: что было характерно для лучших команд, в которых вы работали?**

Майк: Чаще всего лучшие команды, в которых я состоял, имели ясную и возвышенную цель. Это была некая миссия, которую нужно было выполнить и на которой команда была сосредоточена. Когда у вас есть что-то подобное, можно отвлечься от всякой муры вроде политики и спокойно работать.

Эндрю: **Было ли так, что вы понимали цель, а ваша команда нет? Как вам удавалось синхронизировать понимание цели?**

Майк: Что ж, часто команда просто не понимает цель. И задача менеджера проекта, а в Agile-проекте – владельца продукта или скрам-мастера, – убедиться, что команда понимает цель правильно.

По поводу важности цели: многим из нас случалось работать в компаниях, которые ставили целью «заработать кучу денег». Не очень-то возвышенная цель. Нужно, чтобы у команды было что-то еще, кроме подобной задачи.

Нет ничего плохого в том, чтобы заработать кучу денег, но это вряд ли сплотит команду для достижения действительно выдающихся результатов. Да, они будут хорошо работать и из-за денег, но я бы хотел, чтобы продуктивность моих людей шла не от подобной мотивации.

Вот пример. Одна из компаний, в которой я работал, занималась здравоохранением. Собственно, мы первыми придумали сажать медсестер на телефоны, чтобы они давали медицинские советы. Сейчас это обычное дело, но в начале 1990-х ничего подобного не было.

Многие работали в этой компании из-за того, что считали, что мы можем существенно улучшить жизнь людей. Ведь это замечательно – просто позвонить медсестре, вместо того чтобы идти к врачу, и сразу же получить необходимый совет.

Мы выделили типичные звонки и готовили для них ответы. Например, часто звонили молодые мамы с больными детьми, которые не могли обратиться к врачу. После разговора с медсестрой им становилось легче, потому что теперь было ясно, что нужно сделать для своего ребенка.

Мы хотели, чтобы такие звонки стали реальностью для людей. Каждый день мы рассылали по электронной почте запись важного телефонного звонка, который случился накануне. Некоторые из них были спасительными.

Помню, однажды позвонил парень и спросил, идти ли ему к мануальному терапевту. Он повредил спину. У него болело слева под лопаткой, и он не знал, что делать. Идти к врачу? Выпить аспирина и подождать? Оказалось, что это сердечный приступ. Наша медсестра сумела диагностировать его с помощью программ и собственного опыта. Она поняла, что происходит, и вызвала ему домой скорую. Так ему спасли жизнь.

Мы стали рассылать сотрудникам по электронной почте звонок дня, чтобы помочь им осознать, как это важно. Чтобы помочь им понять, почему мы этим занимаемся.

Эндрю: **Да, теперь я вижу, насколько важно для духа команды действительно сплотить людей, дать им почувствовать себя коллективом, а не случайной группой одиночек. Кстати, а сказывается ли это на качестве программ? Мне кажется, вполне может.**

Майк: Думаю, это сложный вопрос. Как тут провести двойное слепое исследование? Просто сказать «Э, да ты сегодня не мотивирован» или «А сегодня как раз да» – и сравнить?

Думаю, есть прямая корреляция между командой с ясной и возвышенной целью и командой, которая пишет качественный код, множество тестов и так далее.

Я всегда говорю: команды, которые работают хорошо, продвигаются вперед быстро. Если вы пишете много хорошего кода, вы многого достигнете, потому что придется бороться с меньшим количеством ошибок.

Команды, в которых я работал, как раз ставили ясную возвышенную цель, и мотивация быстро росла. Так что, полагаю, можно провести аналогию и дальше, предположив, что они и код писали хорошо.

Эндрю: **Мне нравится ваша параллель между быстро сделанной работой и хорошо сделанной.**

Дженни: **Но многим это очень трудно осознать. Чтобы прийти к такому мнению, нужен большой опыт.**

Помню, мы с Эндрю встречали сопротивление, когда хотели донести до людей идею о том, что тестирование и методики помогают лучше писать программы, — им это казалось бюрократией и пустой тратой времени.

Эндрю: **Многие программисты — а также, к примеру, менеджеры — часто не видят связи между кодом и чем-то, не имеющим прямого отношения к коду, — например, что расчеты и планы помогают не просто улучшить код, а улучшить его быстрее, чем без них.**

Майк: Думаю, во многом это происходит из-за того, что мы редко озвучиваем все цели приложения или продукта, который хотим создать. Порой вообще не стоит писать слишком хороший код, это надо признать и в таких ситуациях просто делать работу побыстрее.

Я работаю с некоторыми клиентами, которые делают веб-сайты для рекламных и маркетинговых кампаний. Допустим, мы делаем веб-сайт для фильма, который вот-вот выйдет. И для него не нужно такое же качество кода, как, например, для Microsoft Office. Но если я считаю, что мой код должен прожить 10 лет и больше, то в написание хорошего кода надо вложиться.

Дженни: **Вернемся к вашему первому заявлению о том, что если вы пишете хороший код, то это происходит быстрее, чем с плохим. Как это сочетается с только что сказаным вами?**

Майк: Здесь все зависит от окупаемости. Хороший код – это вложение, и если я имею шансы оправдать это вложение только в течение двух месяцев короткого периода работы веб-сайта, то мои вложения в качество могут просто не окупиться.

Если же приложение будет существовать годами, то вложения, которые я сделаю, бережно и методично проверяя то, что пишу, имеют гораздо больше времени и шансов оправдаться.

Эндрю: **Правильно ли я понял — вы имеете в виду «окупаемость» с той точки зрения, что потом меньше усилий придется потратить на развитие? Думаю, мне это подходит. Вашей команде будет гораздо проще улучшать и поддерживать код, меньше вероятность наткнуться на ошибку, хотя ошибки и неизбежны. За много лет пользователи найдут гораздо больше ошибок, но будут значительно меньше доставать вас, чем если бы им нужно было пользоваться продуктом всего пару недель или месяцев. Так?**

Майк: Именно. Тут есть доля скептицизма. Сейчас я докажу это.

Допустим, вы работаете над приложением, которое уже создали. Вы очень старательны, особенно в отношении качества и понимания кода,

и вам нужно действительно очень высокое качество. У этого приложения две части: само приложение и процедура для импорта данных, которую вы запускаете только раз, чтобы перегнать данные из предыдущей системы.

Очевидно, что любой в этой ситуации напишет более качественный код для приложения, которое должно работать годами. Процедура для импорта запускается только раз. Для нее не нужен такой же строгий подход, как для самого приложения.

Чувствую, что сейчас чуть ли не оправдываю тех, кто пишет плохой код, так что постараюсь выражаться поаккуратнее. Я просто хочу сказать, что в большинстве команд подобный вопрос даже не возникает – а напрасно. Это нужно обсуждать со всеми заинтересованными сторонами: «Как долго должна прожить эта система? Какова наша главная цель?» Иначе мы теряем большие возможности.

Я утверждаю, что если мы пишем эту систему на 10 лет, то я скорее постараюсь повысить качество идеального кода, чем иметь дело с одноразовой процедурой для импорта данных. Конечно, необходимо, чтобы были получены правильные данные, но я все тестирую, проверяю на нескольких образцах данных, все нормально – и готово.

Эндрю: **Знаете, мне это подходит. Меня устраивает, что стандарты качества определяются в начале проекта. Собственно, с точки зрения традиционного управления качеством, разве это не аналог критериев приемлемости? Или определения коэффициента дефектности? Все это есть в учебниках, но ваше свидетельство – это реальность.**

Майк: Думаю, тут нужно поговорить о том, чего обычно не упоминают.

Подумайте о том, как могут изменяться проекты. Не знаю, честно говоря, придем ли мы когда-нибудь к подобной честности, но представьте, что к вам приходит большая шишка и говорит: «Вот мои цели в этом проекте. Я хочу получить новое повышение. Я хочу, чтобы этот проект был сделан, и сделан настолько хорошо, чтобы я получил повышение. Надеюсь, это будет в ближайшие полгода или год. Потом я ухожу совершенно в другое подразделение и плевать хотел на это».

Мы знаем побудительные мотивы этого человека. Не говорю, что это обязательно плохо. Такие побуждения формируются в некоторых компаниях. Но вот так некоторые думают о приложениях, и поэтому команды иногда вынуждены писать код более низкого качества, чем им хотелось бы.

Дженни: **Как при этом мотивировать команду? Ведь если команда реально нацелена на хорошие программы, не скажется ли низкое качество на общем духе? А если члены команды руководствуются разны-**

ми целями? Если у них разные взгляды на то, что они создают и зачем это делают?

Майк: Приведу конкретный пример компании, в которой я был в Бостоне. Я встретился с ее вице-президентом, который пригласил меня консультировать эту организацию. Первым делом я спросил о том приложении, с которым он просил меня помочь.

Они делали новую систему взамен имеющейся системы контроля рабочего процесса. Прежняя была создана в 2000–2002 годах, там применялся замечательный JavaScript, который сейчас мы называем Ajax. Делали ее какие-то приглашенные консультанты, которые точно были гении, раз сумели все так хорошо рассчитать. Но приложение было очень уязвимым, потому что было построено с помощью Ajax до создания самого Ajax.

Я спросил вице-президента, зачем они переписывают систему, и оказалось, что им просто нужно было стабильное приложение, чтобы не тратить слишком много времени на его поддержку. Это была их единственная цель.

Затем я встретился с командами. Мне были не нужны недоразумения. Я не хотел, чтобы в общении возникли какие-то неясности. Поэтому я спросил их: «Зачем вы делаете это приложение? Какова главная цель этого проекта?» И получил такие ответы: «Надо, чтобы оно работало быстрее»; «Нужна большая пропускная способность, потому что документов стало больше». А кто-то ответил: «Мы переходим на новую технологию, потому что наш технический директор прочел об этом в журнале в самолете и решил, что пора менять технологии».

Ни один из пяти или шести разработчиков, с которыми я говорил лично, не дал правильного ответа. Никто не ответил так же, как вице-президент, который был вдохновителем проекта. Они принимали решения, которые приводили их к тому, что новое приложение имело те же проблемы с поддержкой, что и старое.

В этой компании применялся интересный подход: каждые три месяца они выпускали приложения только для внутреннего пользования. Это делало планирование чрезвычайно простым: «Мы успеем к дедлайну 30 июня? Если нет, то давайте успеем к дедлайну 30 сентября».

И босс сказал: «Что ж, я хотел бы получить программу к 30 июня, но пойму, если она не будет готова к этому времени, потому что сроки несколько сжатые». Команда истолковала это как приказ поторопиться и ускорилась в ущерб качеству, чтобы успеть к 30 июня.

Когда я объяснил это боссу, он схватился за голову.

Эта команда с самого начала не обсудила цели, что привело к абсолютно неверным решениям.

Эндрю: **Забавно, да. Слушая вас, я вспомнил, как Дженни рассказывала о команде, в которой когда-то работала, – там было точно так же. Задачи были очевидны и не связаны с дедлайном, но она это понимала, а команда – нет. Ей понадобилась куча времени, чтобы убедить их пропустить дедлайн и сконцентрироваться вместо этого на качестве. Помнишь это, Дженни?**

Дженни: **Да, в итоге мы просто записали цель черным по белому и убедились, что все ее понимают. Но вы правы, люди часто зацикливаются на сроках, на том, чтобы уложиться в расписание. Забывая при этом о цели, к которой нужно стремиться.**

Эндрю: **Вижу, мы вернулись к идее «ясной, возвышенной цели», о которой вы говорили.**

Майк: Да.

Эндрю: **Так что насчет вопроса Дженни о соотношении качества и командного духа?**

Майк: То есть насколько вопрос качества влияет на командный дух? Я понял вопрос так.

Дженни: **Вопрос был о настроении команды, которая делает программы низкого качества. То есть можно ли хорошо себя чувствовать, делая нечто, мягко говоря, паршивое? Как просьба пожертвовать качеством влияет на команду?**

Майк: Тут тоже надо поаккуратнее, потому что в 99% случаев, если команда считает, что ее заставляют писать плохой код, лишь бы побыстрее, это на самом деле не так. Обычно это результат того, что заказчики и разработчики не сходятся в понимании цели проекта. Я уже говорил, что в большинстве случаев мы должны писать намного лучший код, чем обычно, и один из способов сделать это – осознать, что не все элементы приложения нужно писать на одинаковом уровне качества.

Еще один пример – из той поры, когда я начал нанимать людей для различных организаций. У недавних выпускников колледжа было нечто общее. Они считали, что должны выполнить все задания на высшем уровне. Они хотели поразить меня высочайшим качеством любой порученной им работы.

Поразмыслив над этим, я понял, что это просто привычка. Когда вы в колледже, у вас пять или шесть разных преподавателей. А препо-

даватель не скажет: «Хорошо, я немного повышу тебе оценку, потому что ты в этом семестре очень занят на биологии. Облегчу тебе задачу. Сделай работу на 4, и я поставлю 5».

Каждый предмет оценивается отдельно. Я понял это, давая различные задания. Иногда приходилось добавить: «Сделай это как можно лучше». А иногда – ну, допустим, не при программировании, а при определении поставщиков – я говорил: «Нам нужно выбрать элемент-календарь для приложения. Есть восемь хороших вариантов. Но мы не хотим *ошибиться*. Иди проверь их за пару часов и выбери один».

И выпускник понимает это так, что он должен поработать 20 часов и сделать идеальный выбор. «Чтобы никогда о нем не жалеть!» – думает он. Меж тем я бы предпочел просто хорошее решение и 18 часов, потраченных на что-то другое.

Вот и тот нюанс, который я хочу подчеркнуть: не всегда требуется работа экстра-класса. Иногда у нас даже нет возможности показать этот экстра-класс.

Понятно? Я должен убедиться, что это не прозвучало так, словно я призываю писать плохой код. То есть в моем примере это было нормально, хотя в общем случае это не так.

Эндрю: **Да, понятно.**

Майк: Я хотел бы продолжить насчет командного духа. Мое мнение несколько отличается от общепринятого, но, думаю, это ответ на ваш вопрос. Расскажу об одной команде, в которой я работал.

Я управлял очень крупным проектом в одной из компаний, у нас также был второстепенный проект. Не такой важный, но ведь важен любой проект. Некоторым сотрудникам этого проекта руководство установило странные ограничения.

Например, им запретили как-либо обрабатывать ошибки. Если происходит ошибка, боссу наплевать. Приложение может аварийно завершиться. А боссу наплевать. Это было просто нелепо.

Я прослышал об этом, но был занят собственным проектом, критической важности. В другом проекте я не участвовал, только кое-что знал о некоторых его аспектах.

Несколько ночей меня мучила бессонница, так что я приходил в офис довольно рано. Проект был нервный, и однажды я пришел совсем рано – около пяти часов утра. И вот в пять утра я встречаю там двух главных программистов второго проекта, Джеффа и Донну.

Я спросил: «Ребята, что вы тут делаете?» Сначала они не хотели отвечать. Но в конце концов сознались: «Мы добавляем обработку ошибок

и тестируем. Босс запретил нам делать это в приложении, но совесть нам не позволяет. Мы не можем выпустить программу в таком виде, как он требует».

Все три последних месяца они приходили в пять утра и работали до семи. В семь они шли в бар позавтракать. С семи до восьми они ели и к восьми возвращались на работу, притворяясь, будто только что пришли! Потом они до пяти или шести вечера работали над своим приложением.

Когда я наткнулся на них в пять утра, мы это обсудили. Я был менеджером. Они – программистами. Они сказали: «Помоги нам, пожалуйста. Поговори с нашим боссом, объясни, что происходит. Мы так больше не можем». Как раз здесь речь идет о командном духе, потому что они надрывались вопреки желанию босса, который хотел от них низкокачественного продукта.

Я пошел к их боссу и сказал: «Что вообще происходит? Почему вы не разрешаете им обрабатывать ошибки и тестировать, чтобы написать высококачественный продукт? Я согласен с ними, что это неправильное решение». И рассказал ему о том, что творилось. Он все понял. Вот как было дело.

В то время была своеобразная мода давать проектам имена в честь городов. Начала Microsoft, а за ней и другие компании стали называть свои проекты, как города. У меня был проект Napa – как один из городов Калифорнии. Их проект звался Dodge City – по старому доброму вестерну «Додж-Сити».

Босс назвал так проект из-за того, что решил пойти по стопам Universal Studios: в фильме вы видите город, но это всего лишь фасад. Только передняя часть здания. За ней ничего нет. Ни глубины, ни правды.

Он заставлял их работать над приложением, которое хотел демонстрировать на ближайшей выставке, чтобы побудить людей продлить лицензии на этот товар, поскольку скоро выйдет его новая замечательная версия. Но этой версии не было бы. Он не собирался ее выпускать. Вместо этого он намеревался продать похожий, но более дорогой продукт, над которым работал я.

Разработчики трудились над липовым приложением, фантомной программой! Вот почему им не нужна была обработка ошибок, вот почему это должна быть старая добрая демоверсия в духе Нортона 1980-х.

Команда гробилась из-за того, что на самом деле было не нужно. Не говоря о том, что обманывать покупателей некрасиво, каково было команде!

Эндрю: **Какое разочарование для них! Если бы босс с самого начала все объяснил, поставив команде определенную цель, они бы, вероятно, все поняли и сделали, придав приложению лучший вид и не тратя усилия на попытки заставить его работать. Потому что, если задуматься, с проектом, который должен работать, вы связаны по рукам и ногам. А если он не должен работать, то можно добавить в него такие примочки, которые будут выглядеть потрясающе. Лишь бы боссу было, что показывать покупателям или на что намекать.**

Майк: Верно. Здесь нужно было то, что он мог показать на крупной выставке. Если бы оставалась ошибка, то он просто не демонстрировал бы тот фрагмент или взял бы другие данные, или сделал бы это как-то еще. Он мог взять тщательно обработанную демоверсию и запустить только ее.

Эндрю: **Мне нравится эта история: здесь сочетаются обе идеи, о которых мы говорим, — идея связи времени, качества и командного духа и та идея, что понимание целей и перспектив проекта сильно влияет на качество.**

Это приводит меня к одной из причин, по которым мы очень хотели поговорить именно с вами, ведь вы часто читаете лекции о методах, о том, что программисты делают для улучшения кода, для повышения его качества.

Скажем, у вас есть возможность выбрать один, два, три метода, которые действительно могут обеспечить повышение качества кода и планирования проекта. С чего бы вы посоветовали начать команде, которая находится в застое и не знает, что делать?

Майк: Ну, сначала все просто. Первое, что я требую от команды, — это непрерывная интеграция. Думаю, это лучшее, что можно сделать.

Помню один случай, кажется, еще в 1992 году. Одна из команд, в которой я работал, имела дело с типичными для того времени приложениями на C++, когда из-за ряда внешних причин наши дела пошли в гору.

Мы не так много знали о сетевых решениях, чтобы построить действительно хороший билд-сервер. По ряду причин в наших приложениях мы использовали очереди печати фирмы Novell.

И сейчас это звучит странно, а в то время это была просто фантастика. Мы написали билд-сервер, который отслеживал очередь печати Novell. Это было замечательно. Эти очереди поддерживали все. Мы ставили в очередь печати наши скомпилированные работы, а билд-сервер просто отслеживал их и удалял сборки, в которые было что-то вставлено.

Это был совершенно дурацкий способ сетевого решения, но именно в случае с нашим приложением он имел огромное преимущество. Можно было вставить небольшой сегмент в очередь, и билд замечал его.

Не знаю, почему мы не воспользовались этой идеей для мониторинга системы контроля версий. Прошло еще лет семь или восемь, прежде чем я увидел, что команда делает это.

Это полностью изменило мои взгляды. Поистине изумительная техника.

Итак, мой первый совет – непрерывная интеграция. Думаю, для команд это очень важно.

Эндрю: **Непрерывная интеграция как первый шаг мне нравится, потому что это команда может сделать самостоятельно. Ни один менеджер не будет против, потому что любой понимающий менеджер сможет извлечь из этого пользу. Судя по всему, программисты и так всегда это делают.**

Майк: Да, именно. Тут возразить нечего. Здесь даже не надо идти к своему менеджеру и что-то с ним согласовывать. Просто делай.

Но не будем принижать менеджеров. Я сам долго был менеджером. Но я встречал и такие команды, где сотрудники задумываются: «Как объяснить менеджеру, что нам нужна непрерывная интеграция?» Да не надо ничего объяснять. Как объяснить менеджеру, что ты должен писать код? Да никак. Это факт жизни. Просто работай.

Вы правы, думаю, тут не стоит идти и просить разрешения. Может быть, нужно будет купить какое-нибудь дополнительное оборудование, но если взять обычный старый компьютер, валяющийся на складе, то и этого может хватить.

Второе, что я рекомендую командам, – это применение в той или иной форме автоматического тестирования. Это, разумеется, связано с первой целью – непрерывной интеграцией. Это способ заставить приложение непрерывно работать, по крайней мере здесь мы будем знать, что все интегрировано на уровне сборки. Но автоматическое тестирование, безусловно, предпочтительнее. И для многих команд это проблема.

Неважно, на каком уровне тестировать – на блочном или многоуровневом, какими инструментами пользоваться – чем-то вроде FitNesse или системами с пользовательским интерфейсом.

У меня есть некоторые предпочтения по фазам тестирования, но важнее всего просто где-то начать. Часто проще всего начать с высокоуровневых тестов пользовательского интерфейса. Автоматизируйте регрессионный критерий, который иначе придется постоянно запускать вруч-

ную. Автоматизируйте это на уровне пользовательского интерфейса, а затем начните работать над добавлением других видов тестирования.

Эндрю: **Мне нравится то, что вы сказали о коде, билдах и тестировании качества. Но что насчет планирования? Оно может серьезно влиять на проект, но доказать это команде или боссу — или обоим — может оказаться на редкость сложно.**

Майк: Я написал книгу о планировании, потому что мне надоели Agile-команды, которые постоянно заявляют: «Мы гибкие. Мы ничего не планируем». На заре Agile-идеологии, где-то до 2004 года, это было общепринято.

Меня это беспокоило, потому что по своему опыту вице-президента в паре компаний я помнил, как раздражало, если вдруг приходила команда и предлагала процесс, о котором я ничего не знал, рассказывая, какой он замечательный. «Он быстрее, лучше, и покупателям нравится», – и все такое замечательное, что мы часто слышим от Agile-команд. Но потом они наносили решающий удар: «Но надо отказаться от всякой предсказуемости».

Я раздражался, потому что меня искушали отличным процессом, который я не мог использовать. Неважно, насколько хорош процесс. Если он непредсказуем, я не могу на него согласиться.

По опыту работы во многих командах я знаю, что Agile-команды могут планировать. В итоге я написал об этом книгу, просто чтобы развеять миф «Мы Agile-команда. Мы не планируем».

С тех пор я не слышал подобных аргументов, так что, надеюсь, тут я преуспел. Думаю, это просто был вопрос времени и наличия команд, доказавших успешность такого подхода. Планированием занимается множество Agile-команд. Наверное, это играет свою роль: когда вы видите, что другие Agile-команды тратят время на планирование, трудно признать, что вы так не можете.

Я, разумеется, предпочитаю, чтобы команды составляли план. Но вы спросили о двух вещах, которые хочу от команд.

И первая из них обычно не планирование. Надо сначала присмотреться друг к другу, и уж затем можно начинать беспокоиться насчет чьего-то мнения о возможностях планирования.

Дженни: **Хороший подход. В начале интервью вы говорили, что непонимание цели — проблема, сильно мешающая команде быть всегда в курсе того, что она должна сделать.**

Интересно, что все методы, которые вы указали, касались выполнения проекта и создания кода, а не понимания цели проекта. Как вы

считаете, можно ли что-то делать в этом направлении? Можете ли вы порекомендовать здесь какие-то методы?

Майк: По улучшению планирования? Да, конечно. Многие из них описал Джим Хайсмит в своей книге «Agile Project Management» («Управление Agile-проектами») (Addison-Wesley). Он рассказывает о том, что команды должны написать так называемый «лифтовый ответ». Это два предложения, которыми можно ответить встреченному в лифте незнакомцу, который спросил, чем вы занимаетесь.

Помимо этого, я, например, заставлял команду писать обзор продукта, который она хотела бы видеть в журнале по завершении работы.

Я поступил так в компании, которая делала программы для борьбы со шпионским ПО. Обзор их продукта в одном журнале утверждал, что в своей области эта программа находится на втором месте. По мнению редактора, ей следовало вручить второй приз. Одно дело, если мой фильм стал вторым фильмом года: я буду счастлив, я сделаю кучу денег. У вас второй фильм года. Люди смотрят больше одного фильма. Но вторая программа для борьбы с шпионским ПО? Покупают только одну – первую.

Быть вторыми в подобной нише – нет, это не годится. И они серьезно обдумали изменения в процессе. Я попросил их скрам-мастера Эрин обязать команду написать обзор для журнала, суммирующий то, что они бы хотели прочитать о своем продукте через полгода.

Это и была ясная возвышенная цель. И это сработало. Они написали журнальный обзор, и через шесть месяцев вышел настоящий обзор. Помню, что это был Хэллоуин и я сидел дома. В тот день я никуда не ездил и не работал. Я поиграл с детьми, пришел домой и в одиннадцатом часу вечера перед сном проверил электронную почту. Там было письмо от Эрин из трех слов: «Мы сделали это!»

Через неделю, когда я прочел настоящую статью в журнале, я заметил, что некоторые предложения очень напоминали те, которые написали сотрудники в своей версии обзора: «Находит больше шпионских программ, чем любой другой продукт». «Устраняет шпионские программы, которые другие продукты не могут даже обнаружить». Эти и подобные предложения помогли сформировать цель: за полгода до того команда записала их в свой обзор.

Это был завет команды и руководителя, в котором излагалось, чего они должны достичь за эти шесть месяцев, и который осуществился, потому что они постоянно держали в голове осязаемую цель.

Чтобы утвердить такую цель в сознании команды, предлагайте им сделать нечто подобное. Тогда перспективы укореняются в сознании всех людей.

Эндрю: **Еще один вопрос о планировании. Я читал ваши статьи о досках задач и хотел немного поговорить о них.**

И вопрос поважнее: как вы думаете, в чем разница в работе над Agile-проектами и традиционными проектами с проект-менеджерами? Очевидно, что в первом случае нет расписания в традиционном смысле, нет подробных оценок, нет того, что обычно ассоциируется с пыльными кипами пожелтевших бумаг. Но во многих Agile-проектах применяется доска задач — интерактивная доска, по-видимому, эффективный инструмент планирования — в том смысле, который предпочитают программисты.

Здесь я, можно сказать, разрываюсь, потому что за несколько лет добился многих успехов именно благодаря пыльным папкам с планами и расписаниями проектов. Но не могу отрицать, что доски задач и другие инструменты гибкого планирования оказались очень успешными. Итак, включили бы вы использование доски задач в те два-три метода, немедленно взять которые на вооружение требовали бы от команды? Расскажите нам немного о том, как вообще работает доска задач.

Майк: Доска задач (task board) – это то, что я рекомендую всем Agile-командам. Так что считайте, что ответ на ваш вопрос положительный.

Помню, как впервые начал использовать ее в своих Scrum-командах. Я не перенимал здесь чей-то опыт: просто мне пришла в голову идея, которая, как я посчитал, может помочь конкретной команде. Этой команде не давалась концепция совместной работы. Мы провели замечательное совещание по планированию. Мы составили большой список того, что нужно сделать. Но сотрудники не понимали, что они должны быть командой. Они оставались группой индивидуумов.

Эта команда отказалась от традиционного процесса: я предложил им Scrum-методологию. Я взял несколько листов два на три фута и сказал: «Смотрите: вот этот лист будет для одного пользователя, а этот – для второго». И мы приклеили к ним каталожные карточки.

Логики здесь не было. Каждый лист бумаги представлял историю одного пользователя или одной функции, над которой шла работа. Все, что нам нужно было сделать, отмечалось своей каталожной карточкой.

Через неделю я заметил, что дела у этой группы индивидуумов пошли получше, но они по-прежнему не понимали, что должны стать командой. Люди приходили, мы занимались ежедневной рутиной, проводили Scrum-собрания, но на них говорили только о своих индивидуальных задачах. Никто не испытывал интереса к чужой работе.

Мы закончили собрание, я взял все эти листы и прикрепил их к огромной доске, которая была у компании. Распрямив их, я начал делать ко-

лонки: «Вот то, что мы еще не начали: это планы на будущее». Я сделал колонки для текущей и уже сделанной работы. И сказал: «Теперь мы будем передвигать здесь карточки», – и команда поняла идею.

С тех пор я всегда предлагаю это командам. Любая команда, которая целиком располагается в одном здании, на мой взгляд, может с выгодой написать план действий на каталожных карточках и разместить их в произвольно структурированной форме – на доске задач на стене.

Рис. 11.1. *Типичная доска задач, все элементы подписаны. (Публикуется с разрешения Клинтона Кейта)*

Доска задач обрисовывает те шаги, которые необходимо предпринять. Когда оказывается, что, например, дизайн, кодирование или даже тестирование, да что угодно еще не начато, это делается. Эта доска задач примерно так же сложна, как и любая другая, но я видел команды, которые пошли в этом отношении еще дальше.

Эндрю: **Мне кажется, здесь больше внимания уделяется не тому, что нужно сделать через неделю, две недели, три недели, месяц, два месяца, год и больше, а тому, что надо сделать прямо сейчас. Это помогает сразу понять, на какой стадии сейчас находится проект.**

Майк: Думаю, отчасти так и есть. В идеале члены команды, проходя мимо, смотрят на доску и думают: «Что мне важнее всего сделать прямо сейчас? Из всей этой работы что самое важное для меня?»

Мы расписываем доску задач не на шесть месяцев. Планирование задач обычно осуществляется с двух- или четырехнедельным интервалом. Доску задач полностью освобождают, а затем она снова начинает заполняться.

Она необязательно должна иметь сложную структуру. Помнится, впервые начав делать доски задач, как традиционно обученный проект-менеджер, я очень боялся, что только сам смогу увидеть критический путь. Непременно будет критический путь, и никто не сможет его разглядеть, кроме меня, проект-менеджера.

Рис. 11.2. *Доска задач с каталожными карточками на магнитах. Слева то, что предстоит выполнить («Оставленное на потом», «Ошибки», «Разное»), сверху – состояние задач («Сделать», «В работе», «Готово»).*
(Публикуется с разрешения Лайзы Оуэнс)

Я беспокоился по этому поводу первые месяцев шесть, наверное. И через некоторое время до меня дошло: каким бы умным ни был я сам, моя команда коллективно все равно гораздо умнее меня одного. Если на доске будет отражен критический путь, его увидят. Так что больше я по

таким поводам не волновался. До сих пор я еще не встретил команду, которая в последний день пришла бы в ужас: «Боже мой, времени немного, а тут линейная зависимость действий, нам капут!» Ни одна из моих команд не допускала подобных ошибок.

С помощью краткосрочных досок и небольших команд мы сумели исключить целый класс проблем, с которыми иначе неизбежно бы столкнулись.

Дженни: **Здесь мы возвращаемся к вашему рассказу о том, как вы разочаровались в Agile-командах, предупреждающих о непредсказуемости. Теперь ясно, как вы предлагаете осуществлять краткосрочное планирование. Вы вывешиваете задачи на ближайшие три недели, или четыре недели, или шесть – до следующего рывка, до следующего цикла. Как вы при этом предоставляете своему начальнику информацию о планах на ближайшие полгода или год?**

Майк: Безусловно, необходимо составлять план и на шесть месяцев. Можно даже что-то запланировать и на ближайшие год или два. Но суть в том, что здесь требуется уже меньшая точность.

Сегодня у нас 22 октября. Ни один проект-менеджер сегодня не скажет: «Итак, мы назначили дату выпуска. Это 16 сентября следующего года». Даже представить не могу ход вещей, при котором это было бы оправданно. Почему не 15 или 17 сентября? Такой уровень точности просто не нужен на длительный срок.

Сейчас мне опять придется быть поаккуратнее. Я не говорю, что нельзя назначать дату 15 сентября за 11 месяцев. Но если так, то должна быть гибкость. Мы не можем наперед ограничивать себя 11 или 12 месяцами. Немного гибкости – и бизнес пойдет удачнее.

Если нам нужен проект на ближайшие 12 месяцев, надо знать несколько вещей. Я должен узнать у команды, сколько примерно времени это займет, а у менеджера продукта или маркетолога – сколько денег мы собираемся заработать.

Но мне ведь не нужна точная сумма ожидаемых доходов до последнего доллара или евро, мне не нужно подробное расписание до последнего дня. Если в этих оценках будет меньше точности – это нормально. Мы по-прежнему сможем принять соответствующее решение, а это все, что нам нужно.

Эндрю: **У меня еще один вопрос. Мы с Дженни часто выступаем с лекциями и семинарами по всему миру, мы разговаривали со многими представителями индустрии ПО. И я заметил небольшую группу усердных Agile-адептов. Вы, наверное, тоже их заметили: это те, для кого Agile-методы – универсальный ответ на все вопросы.**

Лично я не верю в панацею от всех проблем, связанных с проектами. Я согласен, что многие Agile-команды применяют прекрасные методы, способные помочь многим командам, в том числе традиционным, создавать высококачественное ПО. Вы привели отличные примеры таких методов. Я верю также, что все мы, включая меня, должны быть восприимчивы к новым методам и всегда искать пути улучшения разработки программ.

Но небольшая группа приверженцев и евангелистов, насколько я понимаю, постоянно утверждает, что есть только один способ создать хорошую программу и что все в мире должны изменить своим привычным способам, даже если они работают прекрасно.

Я заметил, что вы так не поступаете. И я хотел бы спросить: замечали ли вы то рвение, о котором я говорю? Как вам удалось не поддаться ему? Может, вы считаете, что я не прав? То есть агрессивная проповедь Agile-методов — вплоть до отрицания всех остальных способов создания программ — является шагом вперед, так что все должны ею заниматься? Понимаю, что вопрос трудный, но думаю, что вы меня поняли.

Майк: Мне кажется, подобное рвение никуда нас не приведет. Да, я считаю, что Agile-подход – это правильно. Я его ярый защитник. Но от излишнего рвения в этой защите меня удерживают две вещи.

Одна состоит в том, что я сам всегда делал программы именно так. На первой своей серьезной работе в компании (не говоря о совсем маленьких компаниях) я делал программы в подразделении Andersen Consulting, которое называлось «Служба информации по судебным процессам». Мы работали над исками. Мы работали над проектами для физических и юридических лиц, которых в чем-то обвиняли.

Вспоминаю, что работал и над громкими делами. Когда имеешь дело с подобными проектами, у тебя нет месяца или двух на то, чтобы что-то заработало. Адвокат приходит к тебе и говорит: «Нам нужна программа, чтобы решить такую-то задачу. Прямо сейчас». Циклы там были очень короткими.

Мне повезло, я знал, как делать программы, еще в 1980-е. Я приобщился к Agile-методологии по счастливой случайности и благодаря обстоятельствам. Сама природа бизнеса, в котором я работал, способствовала этому.

Вторая вещь, которая повлияла на меня, – это то, что я пережил объектную революцию. Я считаю, что Agile-методы – это вторая волна объектно-ориентированной (ОО) революции. Это логическое продолжение той же идеи. Вот почему среди основателей Agile-движения так много сторонников ОО. Они продолжают то, что началось с объектов.

Я не слышал, чтобы люди говорили: «Мне сегодня нужно идти на собрание по ОО-дизайну». Говорят просто: «Мне нужно идти на собрание по дизайну». Объекты победили; никто не говорит больше об ОО. Никто не спрашивает: «А это структурный или объектно-ориентированный дизайн?» Объекты победили, как и полагалось. Большинство команд предпочитает объектно-ориентированную разработку. (Знаю, что к некоторым приложениям это не относится и относиться не может).

И я бы хотел, чтобы Agile-методы постигла та же судьба. Я бы хотел удалить слово *Agile*. Я предпочитаю больше не говорить об Agile-разработке программ. Я бы хотел говорить о разработке программ. Не знаю, сколько времени это займет – лет пять или десять, но в конце концов мы перестанем употреблять это слово: оно станет совершенно естественным.

Agile-методы соотносятся с проектами в большей или меньшей степени, как одни проекты работают с объектами лучше других. Нам не нужны книги со словом «Agile» на обложке. Нам не нужна конференция по Agile-программированию. Пусть это будет просто конференция по программированию.

Agile-идеи становятся общепринятыми и побеждают. Мы просто начинаем именно так все и делать.

12

Как борцы против авторских прав захватили Мордор

Кори Доктороу

У наших возможностей всегда есть определенные пределы. Попробуйте выбрать ресторан для ужина так, чтобы с вашим выбором согласились десять друзей, или составить план свадебной церемонии, который устроит всех, и вы столкнетесь с жесткими ограничениями, налагаемыми коллективным действием. Если вы переходите на уровень компании среднего уровня или большой семьи, становится понятно, что большая часть времени у вас уходит не на то, чтобы делать что-то, а на то, чтобы выяснить, как это можно сделать.

Можно назвать это «операционными издержками»: платой за выполнение чего-либо, накладными расходами при получении общего согласия в вопросе выбора ресторана, пути развития или общей позиции. Эта стоимость, в случаях когда задействованы большие группы людей и объемы работы, может быть так высока, что на многое мы даже не замахиваемся («В эти выходные мы с друзьями планируем свернуть горы!»).

У этих издержек много названий, можно называть их *накладными* или *волокитой* – но, в конечном итоге, единственно верные названия для них – *чушь*, *дурь*, а то и что покрепче. Например: «Я никогда не смогу убедить в этом весь офис – вы, вообще, представляете, сколько <...> мне придется для этого разгрести?»

Как общество мы можем себе позволить делать только то, польза от чего превышает объем разгребаемого. И чем больше проект, тем большая часть его стоимости будет приходиться на подобную чушь.

Но Интернет может побеждать такие издержки.

ВОИС (WIPO, World Intellectual Property Organization) – это Всемирная организация интеллектуальной собственности при ООН. ВОИС олицетворяет все ужасы, с которыми сталкивается Интернет: закрытое заседание в Женеве, зал, набитый дипломатами, на которых жестко давят представители крупных патентных бюро и компаний, занимающихся авторскими правами, а финансируются такие заседания большей частью за счет тех, кто подал заявку на регистрацию патентов, а это значит, что лоббирующие стороны представляют интересы тех, кто оплатил помещение, в котором заседают эти дипломаты. Именно здесь рождается огромное количество подобной дури.

Практически все неудачные и неработающие правила и положения, относящиеся к Интернету, можно проследить до момента их зарождения в ВОИС. ВОИС по отношению к этим дурным правилам – то же, что Мордор по отношению к злу, это бездонный источник страшной глупости, за которой стоит мощь одной из самых влиятельных организаций из всех, когда либо существовавших, – самой Организации Объединенных Наций.

Но в 2006 году многомиллионный и рассчитанный на много лет проект по созданию нового беспощадного Соглашения ВОИС пал под натиском горстки активистов, которые при помощи Интернета зарубили эти переговоры и добились принятия другого соглашения, защищавшего общее право на доступ и распространение человеческих знаний.

Соглашение о вещании было попыткой создать новый уровень дурацких правил для формата видео – правила, подобные правам на авторскую собственность, на использование материалов вещания. Идея была в том, чтобы создать новое право на содержимое вещания, отдельно от авторского права, согласно которому оно должно было принадлежать вещательной компании.

Например, вчера вы записали на видео заседание местного городского совета, где обсуждались проблемы финансирования школ, и поймали члена совета на нарушении своих предвыборных обещаний – у вас на холодильнике до сих пор висит листовка, где он утверждает, что не пойдет на сокращение расходов на школьное образование и точка. Чувствуя себя оскорбленным, вы решаете выложить ролик на YouTube, сопроводив его собственной речью о том, как это обещание склонило ваш голос в пользу этого кандидата в совет.

Согласно авторскому праву вы вправе так поступить. Короткий клип с целью комментария политической ситуации, пародии, аналитического обзора, в образовательных целях и ряда других защищенных целей пользования подпадает под юридическое определение «допустимого добросовестного использования».

Но если бы было принято Соглашение о вещании, закон об авторском праве стал был неприменим, поскольку помимо авторского права на ваше произведение существовало бы «право вещания», которое контролировалось бы вещателем. И правила допустимого добросовестного использования в отношении прав вещания отличались бы от авторского права, а значит, вам потребовалось бы разрешение вещателя в случаях, где оно никогда не требовалось прежде.

Таким образом, Соглашение о вещании перекрывало бы то, что возможно согласно авторскому праву. Но оно шло еще дальше. Право на вещание применялось бы также к материалам, на которые *невозможно* распространить авторское право. Под авторское право подпадает только творческая деятельность, а не фактическая информация. Во многих странах запрещено облагать авторским правом результаты труда правительственных чиновников (если вы оплачиваете этот труд путем перечисления налогов, с какой стати вам нужно получать отдельное разрешение на использование его результатов?). Все эти произведения внезапно оказались бы собственностью компании, которая занимается их вещанием.

Но, что наиболее важно с точки зрения Интернета, вещательное право распространялось бы и на те случаи, когда создатели произведений сознательно желают распространить свои произведения. Огромное количество вещей в Интернете было создано для того, чтобы ими поделиться, чтобы их можно было копировать и использовать. Проект под названием Creative Commons начал обеспечивать получение бесплатных лицензий на публикацию произведений, и в замечательно короткие сроки (с 2003 года) более *160 миллионов произведений* было выпущено под этими лицензиями, опубликовано их создателями, которые желали, чтобы публика могла «копировать, смешивать и записывать» эти вещи с минимумом проволочек и бюрократии. Миллионы слов содержимого Википедии управляются подобной лицензией, лицензией свободной документации GNU, которая также позволяет бесплатное распространение свободной энциклопедии.

Но согласно Соглашению о вещании компания, которая занимается вещанием (или интернет-трансляцией) видео, лицензированного в рамках Creative Commons, могла бы запретить вам, аудитории, копировать, компилировать и распространять это видео *вне зависимости от того, что создатель этого видео разрешает такое его использование.*

ВОИС – это огромная неповоротливая махина. На ее заседаниях присутствуют представители правительств, многие из них представляют небогатые страны, которые не могут себе позволить проживание в Женеве сотен экспертов и участие во всех переговорах разнообразных комитетов при ООН. Большинство делегатов в ВОИС не являются экспертами по

авторскому праву, творческой деятельности или экономическим вопросам, особенно если они представляют не очень богатую страну. Весьма вероятно, что они могут оказаться специалистами в области сельского хозяйства или здравоохранения, то есть, по вопросам, наиболее важным для африканских стран.

Выходит, что эти правительственные делегаты полагаются на мнение «технических специалистов» из так называемых «неправительственных организаций». Большинство людей думает, что сочетание НПО относится к организациям вроде Красного Креста или Международной Амнистии, однако в ООН этот термин означает просто любую организацию, не принадлежащую правительству. В ВОИС большинство таких НПО – это ассоциации кинопроизводителей, вещателей, фармацевтических компаний, звукозаписывающих компаний, издателей – то есть тех, кому выгодно принятие соглашений, представляющих большие права компаниям – обладателям авторских прав и патентов.

Эти экспертные консультации являются неотъемлемой частью работы ВОИС и именно на них основывается решение при принятии соглашений, поэтому вполне ожидаемо, что каждое новое соглашение ВОИС отнимает все больше прав у пользователей и наделяет все большими правами компании.

Но в 2003 году все изменилось. Тогда полномочия наблюдателей от ООН были даны небольшим коалициям технических НПО, представляющих «интересы общественности», таким как Consumer Project on Technology (группе, возбудившей иск по нарушению антитрестовского законодательства против Microsoft), Electronic Frontier Foundation (которые десять лет вели успешные бои на юридическом поприще по вопросам частной собственности и свободы в сети) и Public Knowledge, организации на базе Министерства обороны США, которая провела большую работу по лоббированию в Федеральной комиссии по связи США по вопросам, связанным со свободой эфира.

Переговоры по Соглашению о вещании на тот момент велись уже годами. Проект этого соглашения был распространен в широких кругах, и крупные игроки – США, Европейский союз и Япония – торопили принятие окончательного решения. Практически всем участникам нравилась идея принятия такого документа, более того: комитету, занимавшемуся разработкой этого соглашения, грозили роспуском в случае, если документ не будет готов в ближайшее время. Все шло к тому, что Соглашение о вещании будет принято и можно будет двигаться дальше.

Группа, представлявшая интересы общественности, практически не финансировалась (а направлять делегации в Швейцарию – слишком большие расходы для таких маленьких некоммерческих групп) и не имела дипломатического опыта ведения переговоров. Первые несколько

часов заседания – которые они смотрели на экране в отдельной комнате, поскольку им не были предоставлены места в основном конференц-зале, – прошли в атмосфере чудовищной скуки. Делегат за делегатом брал слово, чтобы заявить что-то вроде:

> Господин Председатель, я впервые беру слово на этом заседании постоянно действующего Комитета по авторским и смежным правам и хотел бы выразить свою признательность и гордость тем, что нас возглавляет такой человек и твердой рукой ведет нас к скорейшему принятию решения по этому вопросу. Недавно было проведено заседание министров моей страны по данному вопросу, и мы хотели бы заявить, что мы в целом поддерживаем разосланный проект документа, в частности в отношении предложений А, Б и В. Однако нас беспокоит, что включение в проект соглашения интернет-трансляции может помешать скорейшему принятию решения, на которое мы все надеемся, и мы хотели бы услышать от своих коллег о возможности выделения положений по интернет-трансляции в отдельное приложение или дополнение к Соглашению, которое будет вариативным и нормативным, но не обязательным к исполнению...

И так часами.

Чем дольше продолжалось заседание, тем больше активисты чувствовали, как сгущаются тучи. На заседании даже не был обеспечен доступ в Интернет – а есть ощущение, что любое заседание, рассматривающее реформы в Интернете, где ограничен доступ в эту сеть, наверняка принимает решение о его запрете. И тут у группы возникла идея. Венди Сельцер из Electronic Frontier Foundation создала на своем ноутбуке беспроводную сеть и запустила программу SubEthaEdit, позволяющую нескольким людям одновременно вносить правки в документ по сети.

Через считанные минуты все специалисты в комнате подключились к компьютеру Венди и записывали все, о чем говорилось на заседании. Один набирал текст, другой правил орфографию, третий добавлял комментарии к тексту. Классический пример высокотехнологичной борьбы с бюрократией и глупостью: вместо того чтобы раздавать записи, делать в них пометки и затем сводить результаты воедино, все работали вместе и одновременно.

В обеденный перерыв несколько активистов прокрались туда, где располагались терминалы, отключили от одного из них Ethernet-кабель и, подключив его к ноутбуку, загрузили эти записи в Интернет. Активисты разместили их в своих блогах – как первый взгляд внутрь переговоров по авторскому праву ООН.

На следующее утро все эти записи попали на Slashdot.com – невероятно популярный среди гиков новостной портал – и оттуда разлетелись

по тысячам различных сайтов. Миллионы людей читали о невнятных махинациях закрытого комитета в Женеве. Начались звонки делегатам разных стран из всех столиц мира, где их разносили за то, что они поступаются интересами нации, или наоборот, поздравляли с тем, что они занимают смелую позицию, столь непопулярную на заседании. Один из делегатов вспоминал потом: «Мне позвонили из правительства, что в министерство поступил звонок от руководителя крупнейшей технологической компании, который сообщил, что наша страна очень хорошо себя показала в ООН, о чем он прочитал на каком-то „слэш-доте“».

Начиная с этого момента активисты дважды в день публиковали свои записи, где отражали все, что происходило в ВОИС. Эти записи сопровождались комментариями и переводом на доступный язык канцеляризмов, столь популярных в ВОИС. Они высмеивали технически не реализуемые прогнозы делегатов и восхваляли делегатов, отстаивавших свободу.

Активисты не могли позволить отвлекать делегации стран от шикарных обедов или обхаживать их на фуршетах. Вместо этого они ежедневно писали политические меморандумы, где излагали свою позицию и разбивали аргументы другой стороны, а затем при помощи блогеров всего мира за ночь переводили их на несколько языков, и уже утром, до возобновления прений эти тексты были распространены. Администрация ООН приостановила действие положения, по которому НПО могли бесплатно копировать документы, – услуга, которая прежде десятилетиями предоставлялась таким организациям, как Ассоциация кинопроизводителей, фармацевтические исследовательские общества и ассоциация «Производители Америки». Теперь, когда активисты с помощью прессы сделали деятельность ООН прозрачной для общественности, эта возможность была запрещена.

Задолго до этого загадочная личность или личности фактически *крала* распечатки активистов с информационной стойки: пропадали целые пачки документов, позже их находили в туалетах или за цветочными горшками. Секретариат игнорировал просьбы о том, чтобы один из многочисленных охранников в вестибюле присматривал за стойкой – или о просмотре записей с видеокамер наблюдения, с тем чтобы выяснить, кто же систематически саботирует распространение «экспертного мнения» в ВОИС.

Группы представителей правообладателей и чиновники ВОИС были в ярости. Один из представителей таких групп пригрозил подать в суд за нарушение авторских прав, поскольку активисты записали его речь и опубликовали ее без его согласия. Секретариат обвинял активистов в «нарушении правил гостеприимства ООН», угрожая наложить запрет на распространение всех печатных материалов, создаваемых НПО.

Но публикации в интернет-дневниках продолжались. Активисты использовали случайные беспроводные сети для проверки публикуемых друг другом заявлений, превратившись с помощью сети в этакого сверхчеловека, поскольку все помогали друг другу оттачивать аргументы и вместе действовали как единое целое.

Из этих интернет-записей группы активистов по всему миру узнали о том, что происходило в ВОИС, и на каждом заседании появлялось все больше и больше людей. Однажды ко мне подошел один из представителей групп правообладателей, тот самый, который грозился подать на нас в суд за нарушение авторского права, и протянул мне листок с текстом.

«Что это?» – спросил я.

«Это речь, которую я собираюсь сегодня сказать. Я хотел бы, чтобы в ваших записях мои замечания были верно отражены».

Вскоре ко мне подошли делегаты одной из стран: «Знаете, нас совсем недавно перебросили на эту дипслужбу, обычные перестановки в правительстве. Мы пытались почитать официальные записи, но вы знаете...»

Вообще-то, я знал. Официальные отчеты ВОИС выходили через месяцы после заседаний, были написаны невозможно канцелярским языком и публиковались только после того, как каждый «цитируемый» делегат отредактирует свои замечания и уберет из них все, что, по его мнению, не подлежит записи. По сравнению с чтением протоколов ВОИС заучивание наизусть телефонной книги – невероятно увлекательное занятие.

«Ну, мы пытались читать эти расшифровки стенограмм, но в конечном итоге перешли на записи, публикуемые активистами. Знаете – они гораздо понятней и доходчивее».

Пройти путь от изгоя до официального биографа всего за несколько лет! Активисты боролись с попытками ВОИС одолеть Интернет при помощи самого Интернета – используя недорогие технологии сотрудничества и коллективных действий для применения умной и непреклонной силы. Противники *могли бы* использовать Интернет для ответных ударов, но они этого не сделали; ВОИС и все поддерживающие ее мастодонты абсолютно неправильно понимали принципы работы Интернета, и их потуги массового воздействия при помощи Интернета были столь же нелепы, как, скажем, попытка белого босса читать рэп на алкогольном рождественском корпоративе с караоке.

Насколько же мощными были действия активистов, скоординировавшихся при помощи Интернета? Соглашение о вещании было отправлено ВОИС пылиться на полку. В свою очередь, ВОИС предпочла разрабатывать новое соглашение, написанное широкой группой активистов, технических специалистов, библиотекарей и гиков всех мастей, на-

званное Соглашением о доступе к информации. В отличие от всех прочих соглашений ВОИС, в которых прописывалось, что публике *не разрешается* делать в отношении работ, охраняемых авторским правом, Соглашение о доступе к информации устанавливало минимальные информационные права, которые каждая страна обязуется предоставить своим гражданам, такие как права на предоставление информации, выполнение копий в специальных форматах для людей с ограниченными возможностями и на сохранение копий в библиотеках и национальных архивах.

13

В защиту Свободного мира

Интервью с Нилом Сигелем

В мире программного обеспечения не так много должностей, равных по масштабу позиции главного инженера сектора Информационных систем (35 000 сотрудников) корпорации Northrop Grumman[1], одной из крупнейших в мире организаций в области оборонных технологий. Мало кто за пределами оборонного сектора (включая нас) располагает информацией о военно-технических и оборонных проектах. Нил Сигель дает нам уникальный шанс взглянуть на оборонную промышленность изнутри и узнать, какова мотивация людей, работающих в этой отрасли.

[1] Northrop Grumman – ведущая американская военно-промышленная компания в области электроники, информационных технологий, авиакосмической отрасли, судостроении по государственным и коммерческим заказам.

Эндрю: **Расскажите немного о Northrop: как вы туда попали и как строилась ваша работа с командами?**

Нил: Northrop Grumman – одно из крупнейших предприятий оборонной и авиакосмической промышленности США, и объем выполненной за годы ее существования работы весьма значителен. Корпорация приобрела свой нынешний вид в результате ряда имевших место за последние 10 лет слияний компаний с долгой и славной историей. Мы называем себя «компанией иммигрантов», отдавая дань этой истории и достижениям прошлого. При этом мы стремимся получить синергетический эффект от объединения этих знаний и умений, достигнув недоступных прежде высот. В настоящее время я занимаю пост вице-президента сектора перспективных систем и технологий, на этой должности я уже около семи лет. До этого я возглавлял одну из компаний в структуре корпорации – «Тактические системы». «Тактические системы» занимались обеспечением жизненно важных функций вооруженных сил США – системами управления боевыми действиями и системами оперативного управления и контроля, в основном для сухопутных войск, но и для других родов войск. Это была очень успешная компания, рост ее оборотов составлял в среднем 25% в год в течение всех семи лет моего руководства.

Эндрю: **Над какими проектами вы работали?**

Нил: Мы создавали оборудование, обеспечивающее связь и координацию действий военных на поле боя, позволявшее определять местонахождение (своих и врагов), создавать и передавать планы и отслеживать их выполнение. Работа была во многом построена на взаимодействии разных групп; 50% доходов мы получали за счет привлечения поставщиков и субподряда, а это означало, что нам нужно было не только создавать эффективные команды в рамках компании и совместно с нашими заказчиками, но и сотрудничать с многими другими компаниями. Надо отметить, что в то время все типичные договоры по компании предполагали самостоятельное выполнение только 70–80% объема работ; то есть мы всерьез зависели от умения создавать эффективные команды не только из сотрудников разных отделов компании, но и совместно с другими компаниями. Это было очень интересно, хотя и очень сложно.

Эндрю: **Почему это было сложно? В чем именно сложность работы с командами других компаний?**

Нил: Самое сложное – это выстроить общие цели. У каждого есть свой набор задач и мотивировок, и даже в рамках одной компании очень редко бывает так, что у двух разных людей или двух разных организаций

эти цели и мотивировки полностью совпадают. А уж если вы работаете с людьми из других компаний, расхождения в мотивации и целях будут просто огромны. Например, вполне возможна ситуация, когда заключается соглашение о совместном участии в торгах с компаниями, которые, на самом деле, хотели бы выполнить весь заказ самостоятельно; в результате разных причин вам удается уговорить их стать субподрядчиками и делать эту работу вместе с вами, но в течение всего проекта они будут подспудно считать, что могли бы сами выиграть этот тендер и стать генеральным подрядчиком. В любом случае, всегда есть такие факторы, как «мы хотим, чтобы нам больше доверяли», или «нам нужна большая прозрачность в работе», или «нам нужны большие объемы», или «почему она отдает эту работу другой стороне? Мы бы сделали лучше».

То есть все вещи, обычные даже в рамках одной компании, усугубляются, когда вы сотрудничаете с другими компаниями, потому что каждое рабочее задание должно оформляться в виде отдельного договора субподряда либо как партнерское соглашение – каждый аспект взаимоотношений регулируется очень формально. Каждый вопрос решается путем формальных переговоров, а решения фиксируются в виде изменений в договоре субподряда. В каком-то смысле в привлечении внешних партнеров есть свои преимущества, потому что вы вынуждены идти на все эти формальности, позволяющие избежать двусмысленности и неопределенности в тех вопросах, по которым вы пришли к соглашению, а двусмысленность может стать настоящим врагом эффективной работы команды. Двусмысленность – это основной источник конфликта из-за несовпадения ожиданий.

Эндрю: **Видимо, тут нужен очень точный баланс. Похоже, вы шли к этому непростым путем. Не каждому выпадает такой опыт. Расскажите об этом.**

Нил: Позвольте мне вернуться к тому времени, когда я руководил «Тактическими системами». Мы участвовали в тендере и получили крупный заказ на разработку программы для сухопутных войск, это была очень важная программа для ведения боевых действий (ее используют до сих пор), и к моменту, когда мы подавали свое предложение, в нашей команде был уже десяток компаний. Кто-то из них должен был разрабатывать отдельные компоненты программы: это были очень узко специализированные компании с четко определенным объемом работ, поэтому с ними редко возникали конфликты или неопределенности. Но другие участники считали себя универсалами, располагающими всеми возможностями для оказания полного спектра услуг, однако будучи частью такой большой команды им приходилось ограничивать себя парой

конкретных задач. В результате некоторые из этих компаний пришли в проект с чувством, что им пришлось пожертвовать своими амбициями для повышения шансов на успех в торгах, и хотя, объективно говоря, так оно и было, эмоции оставались, и многие чувствовали, что могли бы и хотели бы делать больше. Вот здесь, действительно, возникала возможность расхождения в целях и даже конфликта.

Очень увлекательной задачей было, во-первых, уговорить все эти компании стать вашими субподрядчиками, а не соперничать с вами, подавая собственные заявки на участие в тендере на получение генерального подряда или заключая партнерские соглашения с другими компаниями. Главной задачей было убедить их в том, что их общая ожидаемая финансовая и стратегическая выгода в этом случае выше, поскольку выше вероятность успеха. Возможно, их доля в объеме работ – и ее долларовое выражение – будет меньше, но вероятность получения контракта настолько выше, что общий финансовый пакет в этом случае более для них привлекателен. Иногда убедить в этом людей очень непросто.

После, когда создавалась команда с таким большим числом участников, а в том случае по разным причинам это было просто необходимо, и становилось очевидно, что ни одна из этих компаний не может получить значительную часть объема работ, они начинали спрашивать: «Хорошо, я умею делать вот это, но я умею делать и вот то, почему же эти работы не передают мне? Почему вы отдали их другой компании?» Это непростая дипломатическая задача: нужно объединить всех и задать общую цель для всех (и продолжать поддерживать эту общность!), при этом у каждого нужно создать правильную мотивацию, потому что команда из, к примеру, 10 разных компаний в глазах заказчика должна представлять собой единую команду.

Эндрю: **Мне никогда не доводилось работать с проектной группой, где сотрудничают 10 разных компаний, но я четко понимаю необходимость создания общности и мотивации в команде. Как вы добивались этого?**

Нил: В нашем бизнесе есть свои преимущества и недостатки. Первое, что мы делаем на проектах оборонной промышленности для того, чтобы создать мотивацию и объединить людей, – объявляем о нашей общей миссии: мы стоим на страже Свободного мира. Я был генеральным директором этой компании в период, когда разразился так называемый бум доткома. Этот бум создал огромную потребность в технических специалистах. Поиск талантливых разработчиков для нашей компании стал весьма нетривиальной задачей, поскольку бум доткома с огромной силой притягивал таланты, и зарплаты в этом секторе были на тот момент очень высоки. Тогда я при найме кандидатов активно пользовался

значимостью миссии наших заказчиков. Я говорил: «Вы талантливый программист. Можно устроиться в телекоммуникации, и делом всей жизни будет коммутация звонков. А можно прийти работать к нам; платить мы, конечно, будем меньше – хотя голодать тут никому не приходится, – но вы будете стоять на защите свободного мира".

Эндрю: **Вполне представляю, что это может стать фактором мотивации – не для всех, но, определенно, для тех, с кем вам хотелось бы работать. Сложно было находить таких людей?**

Нил: Это процесс самоотбора. Те, для кого этот фактор был важен, шли работать ко мне. Те, для кого была важна только высокая зарплата, не шли. У меня была такая стратегия: я старался, чтобы зарплата у нас была всего на несколько процентов ниже, чем в большинстве компаний нашего региона. Таким образом, я мог отбирать людей, которым импонировала миссия наших заказчиков, пользоваться естественным преимуществом нашей отрасли: мы делаем важную работу, о которой можно рассказать простыми словами. Кроме того, это интересная работа. По-моему, она во всех отношениях столь же интересна, как работа у операторов связи или в Microsoft, – это субъективно, конечно, но работать здесь, определенно, интересно. Это хай-тек, новейшие технологии. Поэтому я могу играть на целой комбинации преимуществ работы у нас: она *важная, интересная* и *достойно оплачивается*.

То есть люди, которые попали ко мне в команду, прошли самоотбор по этим критериям. И это важно: начальным объединяющим вектором была не компания и не я сам, а идея «так можно помочь обороне нашей страны». Та программа разрабатывалась под очень пристальным вниманием вооруженных сил – мы, например, регулярно встречались с начальником штаба сухопутных войск и так далее. Таким образом, при работе над тем проектом мы могли видеть, что делаем нечто ощутимо важное. Более того, военные хотели осуществить этот проект в кратчайшие разумные сроки. Крупные проекты в оборонной промышленности могут длиться семь, десять лет, а то и дольше; например, чтобы построить ракетоносец, нужно как раз семь лет. Но правительство хотело, чтобы эта новая программа была развернута быстрее, чем обычно. Поэтому мы имели удовольствие не только работать над чем-то важным и интересным, но и увидеть результаты своей работы уже через три года.

Таким образом, у меня была возможность объединить команду на почве этих высоких идей: оборона страны, сухопутные войска Соединенных Штатов, впервые осуществляемый перевод в цифровой формат боевых действий, спасение жизней солдат и моряков. Это давало прочную и понятную основу для обеспечения общности целей в команде. Пола-

гаю, в других отраслях это сложно применить в том же виде, но если подойти творчески, в любом деле есть что-то, что делает его по-своему важным и необходимым.

Эндрю: **Удивительно, но ваши слова во многом совпадают с тем, что я слышал от участников проектов свободного ПО: о чувстве работы во имя высокой цели. В вашем случае это высшая цель, вопрос жизни и смерти, но большинство из нас и в своих проектах может найти какую-то подобную мотивацию. Как вы думаете?**

Нил: Национальная оборона – это очень ощутимая цель, но, думаю, во многих отраслях есть свои важные аспекты. Например, работа в электроэнергетической компании может быть очень важной, потому что критически важно обеспечивать бесперебойное, надежное энергоснабжение. В каждой команде можно задать мысль, что ее работа жизненно необходима для общества. Полагаю, в каждой отрасли есть что-то, к чему можно подойти с этой точки зрения.

Эндрю: **Давайте вернемся к тем 10 компаниям, которые вам нужно было уравновесить. Как вы добились этого? Я понял, каким образом вы мотивировали своих сотрудников, но как вам удавалось добиться того, чтобы все эти компании работали вместе на достижение единой цели?**

Нил: Есть очевидные потребности: людям нужен доход, нужна рентабельность и прогнозируемость. Но, по моему опыту, это очень шаткие основания для объединения людей и целей. Гораздо лучше рассмотреть, каковы стратегические цели компании. Если в компаниях-партнерах вы имеете дело с сотрудниками достаточно высокого уровня, вы можете обсуждать с ними стратегические цели и ваши возможные действия – вопросы, которые могут иметь, а могут и не иметь отношения к краткосрочным доходам, что позволит укрепить возможности, которые будут для них стратегически важны пять лет спустя.

Много времени приходится тратить на то, чтобы понять суть стратегических целей ваших партнеров. Это не так просто, потому что они не горят желанием раскрывать вам все свои секреты, поскольку вы попрежнему их потенциальный соперник по другим проектам. Но попробуйте спросить у них, что им не нравится, каковы их стратегические цели и что вы можете сделать для того, чтобы не только удовлетворить их краткосрочные потребности по зарплате и рабочим местам, но и обеспечить стратегическую пользу от совместной работы. Даже сам факт озвучивания такого вопроса и выслушивания их ответа, вовлечения в диалог, уже создает атмосферу уважения: ощущения, что этот проект осуществляется не только вами и ради вас, или ради прибыли и во сла-

ву вашей компании, но включает также и их компанию. И это, разумеется, длительный, медленный процесс – на основании одного разговора трудно создать доверительные отношения – но это очень важная, ощутимая вещь. Можно найти приемлемые для всех сторон аспекты партнерских взаимоотношений, которые в итоге принесут стратегическую пользу партнерам.

Это верно как для партнеров, так и для других подразделений нашей корпорации. Когда обсуждаешь такие вещи с компаниями в рамках своей же корпорации, это, конечно, немного другое – после переговоров с другими компаниями возникает ложное ощущение полной ясности. Но если видно, что вы пытаетесь выйти за рамки прописанных объемов работ и прибылей и стараетесь, чтобы это сработало в долгосрочной стратегической перспективе, такие диалоги могут способствовать созданию общего чувства удовлетворения от партнерских отношений, даже если непосредственная прибыль будет меньше, чем то, на что они рассчитывали.

Эндрю: **Очень интересно было бы узнать обычный порядок работы в ваших проектах. Полагаю, в начале вам приходится собирать обширные массивы информации, общаться с людьми, которые на практике используют то, над чем вы будете работать. Не могли бы вы рассказать об этом поподробнее?**

Нил: Во-первых, этот сбор информации – замечательная возможность для реализации того объединения команды, о котором я говорил, общности на почве единой миссии. В самом деле, это очень хороший шанс увлечь людей и настроить их на общую волну в отношении миссии будущих пользователей. Командный состав вооруженных сил США представляют, большей частью, очень способные и целеустремленные люди, которые отлично понимают и воплощают свою миссию. Мне очень нравится, что я могу быть соединительным звеном между моими сотрудниками и этими людьми.

Эндрю: **С ними легко работать? Каковы они в работе?**

Нил: Они хорошие клиенты! Они верят в свою миссию и умеют о ней говорить. Конечно, вооруженные силы США – бюрократическая машина еще большего масштаба, чем Northrop Grumman; у нас сотня тысяч людей, а у них – миллион, и в этом есть свои недостатки. С другой стороны, они понимают, что от качества *нашей* работы зависят *их* жизни, причем, в буквальном смысле, а не фигурально – это не свежесть молока или своевременная доставка газет. Их жизнь может физически зависеть от качества нашей работы. Поэтому они очень заинтересованы в том, чтобы предоставить нам как можно более полную информацию:

что они делают, как они делают это, чего они хотели бы. Это дает высокую ясность. Нам нравится, когда наши ребята могут с головой погрузиться в проблемы наших заказчиков. Мы называем это «получением предметных знаний».

Мы всячески приветствуем, когда наши сотрудники, особенно старшие технические сотрудники, становятся настоящими экспертами в конкретной предметной области наших заказчиков. Когда они проникают в суть вопроса, у нас на руках оказываются инженерные знания, технологические знания и знание предмета... и вот тут, по моему опыту, начинается чудо. У нас рождаются великие идеи, которые, в свою очередь, вызывают у заказчиков большое воодушевление и веру в то, что мы действительно можем сделать что-то полезное для них.

***Эндрю:* «Тут начинается чудо» — можно об этом поподробнее?**

Нил: Мне кажется, что в обычном бизнесе – а я работал и в коммерческих ИТ-компаниях и в других структурах – все происходит довольно однотипно. У заказчика есть сотрудники, отвечающие за работу с подрядчиками, те, кто должен общаться, объяснять, что требуется и как они работают. Иногда заказчики становятся сами себе худшими врагами. Они могут неплохо разбираться в том, что делают сегодня, но в будущее они часто смотрят на три дня вперед вместо трех лет, как следовало бы. У них очень узкое понимание своих потребностей.

Мы, например, можем отправить своих старших сотрудников на год в Службу национальной безопасности или в Национальное управление воздушно-космической разведки, или приглашаем офицеров к себе на год, или отправляем инженеров наблюдать за маневрами в Национальном центре подготовки, или даже отправляем своих специалистов в Ирак или Афганистан, чтобы они вместе с войсками патрулировали территорию. Потому что обычно назначенные заказчиком сотрудники для работы с подрядчиками, которые, возможно, хорошо понимают структуру текущей работы, могут иметь или не иметь долгосрочное видение по проекту, скорее всего, плохо разбираются в технологиях... и, следовательно, не очень хорошо осознают все возможности. С другой стороны, в частных компаниях есть технические эксперты, которые хорошо разбираются в технологиях, но если у них нет знания предмета, вам потребуется посредник и переводчик: «Он сказал „а“, но, на самом деле, он имел в виду „б“», – и в этом духе. Мы потратили немало усилий на то, чтобы наши сотрудники смогли стать такими «переводчиками». Но мы поняли, что для того чтобы эксперты в какой-то области стали инженерами или учеными, нужно очень много времени; иногда мы нанимаем военных в отставке, чтобы они могли рассказать нам, как работает организация. Но в то же время я осознал, что гораздо эффективнее

«погружать» наших специалистов в предметную область заказчика, чтобы они получили специальные знания. Тогда мне уже не нужны переводчики, между предметным и техническим знаниями нет больше барьеров. Один человек может совмещать их оба, создавая новые идеи и помогая увидеть перспективу.

Вот к чему мы стремимся – чтобы наши технические специалисты были на высоком уровне предметного знания о работе заказчика. Тогда между технической и предметной сферами нет больше ни фильтров, ни барьеров, и именно в такой ситуации рождаются нетривиальные идеи («1 + 1 = 3») о потенциале будущего. Именно так можно решить проблему недальновидности – вы начинаете работать над тем, как можно применить технологии для создания революционного совершенствования задачи. Вот так мы работаем с заказчиками.

Эндрю: **Есть ли особые методы или инструменты, помогающие вам в этом, навскидку?**

Нил: У нас есть много специальных механизмов. Например, есть практика, которую мы называем «жюри пользователей», когда мы открываем свои идеи. Мы идем к военным, которые будут пользоваться системой, и пошагово рассказываем им, как она будет работать: один день вашей жизни после запуска новой системы. Потом мы обсуждаем, будет ли это полезно и насколько; мы пытаемся оценить проект не с точки зрения технического совершенства, но с точки зрения функциональной пригодности. Есть и много других механизмов; главное – преодолеть или уменьшить барьер между предметным знанием, наукой и проектированием.

Эндрю: **Очень хотелось бы услышать о каком-нибудь конкретном проекте – что вы делали и как применяли эти механизмы на практике.**

Нил: Конечно. Я могу привести пример. При разработке одной боевой системы оперативного управления для военных у нас случился интересный концептуальный прорыв.

Эндрю: **Я не знаком с военными системами – что такое система оперативного управления?**

Нил: Это информационная система для военного командования. Она позволяет им связываться друг с другом и владеть ситуацией, то есть знать, кто где находится и в каком положении. Таким образом, командующий может получить картину происходящего. Прежде поля боя были небольшими по размеру, командующий мог стоять на холме с биноклем и видеть, что происходит. Но примерно к 1900 году поля сражений стали такими большими, что уже невозможно было стоять на хол-

Рис. 13.1. *Солдат в районе боевых действий применяет систему оперативного управления, разработанную командой Нила*

ме, как делал это Наполеон. Появилась необходимость использовать технологии, чтобы «видеть» все поле боя. Система, позволяющая это делать, и называется «системой оперативного управления».

Итак, мы разрабатывали подобную систему. У нас была идея модели, в чем-то похожей на электронную почту: у меня есть информация, которую я хочу кому-то отправить, я ввожу ее в систему и сообщаю компьютеру имя адресата, которому я собираюсь послать сообщение. Одним из удивительных открытий для нас стал очень простой способ создавать такие военные информационные системы; пользователи обычно не хотят знать, кому (по имени) нужно отправить информацию; они хотят, чтобы система определила по характеру самой информации, кто в ней заинтересован, а затем автоматически отправила информацию этому человеку. Если вы находитесь в районе боевых действий, видите что-то и определяете, что это потенциальная цель, – например у сухопутных войск и военно-морских сил есть операция, называемая «вызов огня»: то есть «я хочу, чтобы кто-то подбил для меня вон тот танк». Возможно, вы просто наблюдатель и у вас нет оружия, или у вас нет нужного вида оружия, или вам нужно оставаться нераскрытым и потому вы не може-

те стрелять сами, но хотите кому-то скомандовать это. Но вы не знаете кому. Поэтому вы позволяете компьютеру определить получателя и автоматически отправить информацию по нужному адресу.

Итак, мы в итоге поняли, что они хотят адресовать сигнал «вызов огня» *системе* (а не конкретному человеку) и хотят, чтобы система могла определять, кому должен быть адресован запрос. Потому что вариантов много: можно сбросить бомбу с самолета или пустить небольшую ракету с вертолета, или выстрел может сделать другой танк, или может открыть огонь артиллерия, расположенная в 10 милях от танка. И они не хотят, чтобы решение об этом принимал человек, находящийся в районе боевых действий. Он может сделать неправильный выбор: он не знает, кто находится в районе, кто готов вести огонь, какое оружие лучше использовать. Его задача обнаружить врага и принять решение о вызове огня. А решать, какое оружие или какой механизм использовать, – задача уже других людей.

У нас ушло поразительно много времени на то, чтобы пообщаться друг с другом и понять, что «правильным» подходом будет разработать такую систему, которая сама сможет определять адресата, не заставляя человека, желающего направить информацию, обращаться к конкретным людям поименно. Вот пример препятствий и неоптимальной работы, которые могут стать следствием того, что, пользователь говорит на одном языке, а ученые и инженеры – на другом, и того, как в результате получения специалистами предметных знаний в области применения, обращения к «пользовательскому жюри» и так далее может привести вас к моменту «Эврика! Теперь я знаю, как сделать то, что вам нужно!». У пользователей тоже не было четкого представления об этой системе, поэтому мы вместе долго бились над тем, как система должна работать.

Эндрю: **Чудесная история. Главное для меня здесь вот что: многие думают, что работа в военном проекте связана с огромным количеством бюрократии и бумажной волокиты, но то, о чем говорите вы, видится сложной, но по-настоящему интересной работой.**

Нил: Я не променял бы свою работу ни на какую другую. Я главный инженер организации, где 35 000 сотрудников делают действительно важную и интересную работу. Это замечательное дело для инженера или программиста. Конечно, в компании есть определенная бюрократия, ведь нас 100 тысяч человек, но, с другой стороны, у вас есть 99 999 очень умных коллег. Если вы умеете обратить такие масштабы себе на пользу, можно достичь великих результатов.

Я работал в небольших компаниях. Работать там весело и несложно – если вас всего 20 человек. Но если вам потребуется эксперт в другой

области, возможно, вы не сможете его найти. У нас здесь есть специалисты практически из всех областей, которые могут иметь отношение к нашей работе. С тех пор как я занял пост директора по технологиям крупнейшей корпорации, я знаю этих людей лично либо знаю, к кому из коллег в других подразделениях можно обратиться, чтобы их найти. В этой компании работают тысячи специалистов мирового уровня, и если у нас получается правильно организовать работу, то масштабы работают на нас. Поэтому когда мы в оптимальной форме, это очень увлекательное и чудесное дело для всех. Как у любого бизнеса, у нас бывают и плохие времена. Но для программистов это замечательное место. И у нас есть совершенно ненасытная потребность в разработчиках и инженерах по системам, с одним условием, что почти во всех случаях это должны быть граждане Соединенных Штатов.

14

Спасение жизней

Интервью с Тревором Филдом

Благотворительных организаций много. В организациях, которым можно пожертвовать деньги, недостатка не бывает. Но немногие из них обладают такой эффективностью и новаторским подходом, как PlayPumps International. Сложно представить лучшее применение денег, чем обеспечение чистой питьевой водой отчаянно нуждающихся в ней жителей деревень к югу от Сахары. Уникален и подход PlayPumps: они запускают помпы с помощью каруселей, на которых катаются деревенские дети. Это не только дает питьевую воду, что благотворно сказывается на людях и препятствует возникновению болезней, но и освобождает женщин, которые во многих деревнях вынуждены буквально весь день таскать воду, часто из загрязненных источников. Тревор Филд, основатель PlayPumps, восхищает нас не только потому, что делает великое дело, но и потому, что это дело связано с инновациями, инженерной мыслью и работой в команде. Поэтому часть дохода от продажи каждого экземпляра «Идеальных команд» будет направлена в PlayPumps International. Мы с удовольствием побеседовали с Тревором о его организации.

Эндрю: **Итак, сначала я бы хотел поблагодарить вас за то, что согласились побеседовать. Мы с Дженни очень впечатлены тем, что уже сделали ваши ребята. Поскольку вся наша книга посвящена инженерным командам и тому, как улучшить их работу, мы решили, что вам удалось собрать нечто большее, чем просто хорошо работающая организация; пожалуй, это пример инженерного подвига. Мы хотели бы поговорить с вами о ее становлении, истории и тех проблемах, с которыми вы столкнулись. Если можно, расскажите о том, как начиналась PlayPumps и какие цели стояли перед организацией.**

Тревор: Вода – источник жизни на планете, и я всегда этим очень интересовался. Несколько лет назад ко мне приехали тесть с тещей. Пока жена и теща ходили по магазинам, я взял тестя на сельскохозяйственную выставку в Преторию, в 50 километрах к северу от Йоханнесбурга.

Там был парень – Ронни Стюйвер. Он был бурильщиком, ставил буровые станки, компрессоры и тому подобное. И у него была и модель помпы PlayPump. Она отличалась от того, что мы имеем сейчас, но идею я понял – понял, как это может работать. В этом инструменте было много стальных частей, а цена по сравнению, например, с Afridev или Mono Pump была просто смешная. Я уже давно занимался наружной и журнальной рекламой, и мне понравилась эта идея. Я подошел к нему и предложил дать мне эксклюзивные права на продажу этой идеи. Так все и началось.

Я решил, что лучше всего будет, если мы задействуем частный сектор. Например, закачаем воду в надземную цистерну, а вокруг цистерны разместим рекламу. Сельские жители не большие любители телевидения, Интернета, журналов и газет. В типично сельских районах рекламы нет вообще.

Водокачка – это центр общины. Если бы можно было разместить сообщения, рекламирующие такие товары, которые местные сельские жители обычно покупают, – сахар, чай, кофе, а сейчас и зарядные устройства для мобильных телефонов, – это должно было сработать. Так мы и сделали; начало было положено.

С тех пор все сильно разрослось. Мы купили у Ронни Стюйвера патент и подвергли все переработкам и переделкам. У нас появилась своя фабрика в местечке Моддерфонтейн, рядом с дорогой. На моей фабрике занято 36 человек. Она способна выпускать сотню единиц оборудования в месяц. С этого мы начинали.

Эндрю: **В какой переработке нуждалась помпа? С какими интересными или сложными проблемами конструкции пришлось столкнуться?**

Тревор: Тот парень изготвил модель. Из модели нужно было построить рабочий механизм, что мы и сделали. Вначале несколько механизмов для нас собрал он. Видите ли, Ронни был фермером, и мастерская его была при ферме. Он делал вещи с помощью рук и молотка – обработаны они были плохо.

Так что мы позаботились об оборудовании. Здесь, в Южной Африке, все было в порядке: мы могли производить оборудование сами. Но если хочешь экспортировать, то и продукт должен быть экспортного качества. Итак, мы купили у него патент и переделали все – не с чистого листа, конечно, но вообще каждую деталь нужно было проверить на устойчивость. И мы построили новую модель, в которой были уверены, что ее можно оставить где-нибудь в буше и она там будет работать.

Вот, собственно, и вся главная проблема.

Эндрю: **Эту книгу прочтут многие программисты, которым определенно близка та идея, что начинаешь с собственноручно построенного прототипа, а дальше улучшаешь его, доводишь до ума и в конце концов поставляешь тем, кто станет пользоваться им ежедневно.**

Тревор: Да, именно так мы и поступали. Нужно было отвезти эту вещь в никуда и просто ждать, будет ли она работать. К тому же чувствуешь ответственность перед детьми. Я видел, как на этих каруселях собиралось до 70 детей – один на другом. Средний вес ребенка, катающегося на карусели, примерно 30 килограммов. 20–30 детей по 30 килограммов каждый – серьезная нагрузка. Нужно было убедиться, что устройство выдержит. И что все опоры и движущиеся части действительно крепки и выдержат любой детский вес. Основные опоры могли вынести вес в четыре с половиной тонны – огромная величина по сравнению с действительностью.

Эндрю: **Наверное, дети резвились так, как вы себе и не представляли?**

Тревор: Разумеется. Поэтому мы применили нагнетательный цилиндр. Не буду вдаваться в подробности, скажу лишь, что эта штука ходит вверх и вниз. Дети вращаются, и механизм трансформирует это движение в эквивалентное вертикальное движение, помпа движется вверх и вниз. Есть и вращательные помпы, которые работают по принципу Архимедова винта. Но дело в том, что они работают только в одном направлении: если изменить его, вода перестанет поступать.

А детям только этого и надо, сами понимаете. Они обязательно станут вертеть его в другую сторону, потому что захотят попробовать поразному.

***Эндрю:* Я думаю, это отличная мотивация для всех, кто работает в инженерной отрасли и для тех, кто строит помпы непосредственно на заводе. Ведь на самом деле в мире не так много продуктов, настолько безусловно идущих ему на пользу.**

Тревор: Жаль, что вы не можете приехать посмотреть, потому что мы бы вас позвали на завод. У нас есть рабочие, которые ставят заклепки на рамы и не видят в этом какого-то глубокого смысла. Чтобы мотивировать их, мы развесили везде постеры PlayPumps. У нас была кампания по внешней рекламе – профессиональная, с большими плакатами на дороге. (Мы продали все эти места и избавились от всех своих активов, чтобы сконцентрироваться исключительно на PlayPumps.) Мы напечатали множество постеров с помпами и повесили их на заводе, чтобы люди знали, во имя чего они работают.

А мой партнер – довольно творческий парень, так что помпы раскрашены в красный, желтый, зеленый – очень яркие цвета, как раз как любят дети. И мы раскрасили в эти цвета всю фабрику. Перила у нас красные, желтые, зеленые. Как в фильме «Чарли и шоколадная фабрика»!

***Эндрю:* Одно дело – доставить помпы туда, где их можно собрать и где они могут ожидать ввода в эксплуатацию. И совсем другое – перевезти их в поле.**

***Рис. 14.1.** Завод PlayPumps*

Тревор: Да, это большая проблема. Подумайте только о цене топлива: не знаю, сколько сейчас стоит галлон в Калифорнии, но в Нью-Йорке он, наверное, стоит примерно так же. В Африке транспортировка очень сложна, к тому же наши штуки тяжелые. Очень тяжелые.

Топливо в Южной Африке стоит около доллара за литр, где-то так. Но в Уганде, например, это уже больше семи долларов за литр. С такой разницей ничего нельзя толком запланировать, и это, конечно, мешает логистике.

А в таких странах, как Зимбабве, топлива вообще нет. Это одна из причин, почему мы там не работаем, помимо того что, пока там мистер Мугабе, мы не можем там появиться. Очень жаль, потому что мы бы хотели помочь зимбабвийскому народу. Они страдают, и не по своей вине. Они просто родились в неподходящее время. Это наши соседи. Я могу отсюда съездить на машине в Хараре, и это будет как прогулка к побережью на Рождество. Дороги там хорошие, но таможня требует себе 35% от ввозимого товара. Ну уж нет, только через мой труп.

Эндрю: **Думаю, в работе с другими странами и их правительствами и пограничными службами свои сложности.**

Тревор: У нас есть льготы на ввоз товара во все страны, с которыми имеем дело, потом что получаем средства из Америки, Британии, со всего мира, мы делаем эти помпы и передаем их нуждающимся в них общинах той или иной страны.

Мы настаиваем на том, чтобы получить освобождение от таможенных пошлин. Освобождение от налога на ввоз – одна проблема. Другая – это переезд через границы, потому что у таможенников всегда бывают странные идеи насчет того, что можно ввозить и вывозить.

Есть, конечно, и другие проблемы. Не во всех странах хорошая система дорог. Возьмем, например, 18-колесную двухприцепную фуру с двумя контейнерами по 40 и 20 футов длиной. Мы называем ее интерлинком. Но взять интерлинк в Мозамбик нельзя – они там запрещены. Как и в Зимбабве, потому что слишком портят дороги. В интерлинке можно было бы перевезти 25 помп, а так приходится ограничиваться 11 штуками в обычном фургоне. Следовательно, приходится отправлять две машины, что удваивает стоимость.

Мы только что заключили договор с Малави: я подписал с их правительством Меморандум о понимании. Они с радостью приняли наши помпы и освободили нас от таможенных пошлин. Но чтобы попасть отсюда в Малави, надо ехать через Мозамбик или Зимбабве. И это настоя-

щая проблема: топливо по дороге – редкость, и цены колеблются очень сильно.

Эндрю: **Я как раз размышлял об административных и бюрократических проблемах, с которыми программисты сталкиваются по работе. В следующий раз, обнаружив себя стонущим из-за того, что Боб требует отчет по проекту, я напомню себе, что мне не нужно добиваться Меморандума о понимании от какого-нибудь правительства. Вам вероятно, приходится преодолевать кучу волокиты и языковые барьеры?**

Тревор: Да. Правда, надо отдать должное тем, с кем я общаюсь: большинство из них говорит на очень хорошем английском, ведь они занимают должности в министерствах. Но все равно случается и взаимное непонимание.

Многое приходится объяснять на пальцах, вы вынуждены присутствовать там физически, отнимать свое и чужое время, рассказывать обо всем процессе, о принципе работы, о том, куда мы собираемся поставить эти помпы, кто будет за них отвечать, когда они уже прибудут в страну. Мы должны убедиться, что это безопасно, потому что не хотелось бы позже обнаружить какие-то непредвиденные мелочи. Все части машины собраны на нашем заводе, каждый болтик и гвоздик посчитан и пересчитан и уложен в нейлоновый мешок, где хранится весь крепеж. Иначе только представьте, как установщики, пытаясь поставить помпу в 300 или 400 километрах от Дар-эс-Салама в буше, вдруг обнаруживают, что в мешке не хватает двух болтов.

И это составляет некоторую операционную проблему. Приходится посылать запасные экземпляры и запасные части для установщиков. Модели для каждой страны в основном совпадают с южноафриканской.

Эндрю: **Работа главным образом выполняется в ЮАР? Как функционирует установка? А поддержка? Наверное, это тоже проблема.**

Тревор: В Южной Африке девять провинций. Мы работаем в пяти: Восточный Кейп, Квазулу-Наталь, Мпумаланга, Лимпопо и Северо-Западная. Эти провинции исторически наиболее бедны ресурсами.

В каждой провинции у нас есть своя команда установщиков. Например, мы отсылаем грузовик с 25 экземплярами в Квазулу-Наталь. Обычно он приезжает к муниципальным постройкам, где мы храним оборудование. Наш установщик в Квазулу-Наталь каждый день заезжает, забирает пару помп, едет на место и начинает установку.

Эндрю: **Поддержка работоспособности этих помп, вероятно, тоже интересный вопрос. Нужен ли творческий подход, чтобы справиться с этими проблемами? Как это все работает?**

Рис. 14.2. *Установка помпы PlayPump*

Тревор: Кода установка завершена, установщик превращается в специалиста службы поддержки. На постаменте цистерны мы раньше размещали наш бесплатный номер, начинавшийся с 0800. Если помпа ломалась, этот номер можно было записать на листке бумаги и дойти до телефона в магазине, позвонить в наш офис в Йоханнесбурге и сообщить, что помпа вышла из строя.

Но идея оказалась крайне неудачной. В реальности происходило следующее: помпа ломалась, парень, который записывал номер и звонил в ремонт, говорил нам об этом. Мы спрашивали: «А какая это помпа?» Он отвечал: «Не знаю, ну, та, что возле школы». Мы спрашивали: «Возле какой школы?» А он: «Школы Амбабулла». Мы смотрели по своей базе данных и никакой Амбабуллы не видели, потому что тут для каждой местности семь разных названий.

И это была настоящая проблема, с ней действительно пришлось бороться всерьез. Наконец, мы решили пользоваться SMS-сообщениями. Потому что неважно, насколько глухая провинция и насколько бедны там люди: у кого-то обязательно найдется сотовый телефон. Даже удивительно.

И теперь у нас на каждой цистерне есть номер. Правда, номера цистерн для идентификации были там всегда: например в Квазулу-Натале номер, скажем, KZN001. Это уникальный номер для данной конкретной

Рис. 14.3. *Дети играют на новой помпе PlayPump*

помпы, которая внесена в нашу базу данных и высвечивается на GPS, так что ее можно рассмотреть с 5 метров; видна она и на Google Earth – тоже большое преимущество.

И мы написали такое сообщение: «Отправляйте SMS с номером помпы – KZN001 – на этот телефонный номер». Таким образом, мы видим номер помпы в сообщении и можем ее сразу же идентифицировать. Потом мы можем перезвонить этому человеку, потому что на нашем телефоне отображается его номер, и пообщаться с ним, чтобы убедиться, что помпа вышла из строя.

И это работает гораздо лучше. Казалось бы, простая вещь, но, скажу вам, над этим пришлось потрудиться.

Эндрю: **Иногда простые, элегантные решения сложных проблем и есть самые трудные. И все получилось?**

Тревор: Да-да, очень хорошо. К тому же вы сразу узнаете о том, что помпа сломалась, потому что это ведь источник воды. И по этому поводу звонят не один или два раза: звонки следуют постоянно.

Эндрю: **Помимо организации, которая поставляет в деревни помпы для воды, вы руководите и благотворительной организацией. Уверен, что здесь вы столкнулись с совершенно другими проблемами, чем при основании PlayPumps. Но было ли что-то общее?**

Тревор: Нет, мне никогда раньше с таким сталкиваться не приходилось.

Эндрю: **То есть пришлось многому научиться?**

Тревор: Это случилось в 1999 году, восемь или восемь с половиной лет назад, когда мы участвовали в полностью коммерческом предприятии под названием Roundabout Outdoor, которым занимаемся и по сей день. Мы продавали рекламу на всех четырех сторонах цистерн ради коммерческой выгоды. Запустились мы на юге Йоханнесбурга в местечке Рейтфонтейн. На открытии был бывший президент Нельсон Мандела: это вызвало ажиотаж, ведь он привлекал тысячи людей повсюду, где бы ни находился. Он пришел посмотреть на нашу помпу, как и многие другие: представители Всемирного банка, ЮНИСЕФ, CARE, «Плэнет интернешнл» – множество самых разных людей.

Я разговаривал с представителем Всемирного банка доктором Россом Полом, который теперь вышел на пенсию и живет в Австралии. Я спросил его: «Как вы думаете, где бы нам достать средства для этой системы?» И он ответил: «Есть такой конкурс – Конкурс развития сферы торговли. Почему бы не принять в нем участие?»

И мы второпях подали заявку: уже наступал дедлайн. Я пришел к нему в офис и рассказал о том, что мы сделали, он занес это в компьютер и мы подали онлайн-заявку. Мы попали в шорт-лист, и я поехал на презентацию в Вашингтон, округ Колумбия. Мы победили в конкурсе. Это принесло нам... кажется, в те дни это было 165 000 долларов. Эти деньги мы потратили на установку новых помп. Так все и началось.

Но самое интересное стало происходить, когда нашу идею оценили другие. Нам больше не нужно было погашать средствами от рекламы стоимость оборудования: деньги полились к нам нежданно-негаданно. Один парень из Ирландии дал нам 10 000 фунтов, кто-то еще прислал деньги почтой, другие – банковскими переводами. Мы стали коммерческой организацией, которая принимает пожертвования. Ситуация довольно неуклюжая.

Я поговорил в это время с представителем Министерства воды по имени Ронни Касрилс и спросил, что, по его мнению, нам следует сделать. Он обратился с этим в Министерство финансов, между ними завязалась переписка. Затем меня вызвали в SARS – это наша внутренняя налоговая служба. Со мной чуть инфаркт не случился, когда мне позвонили оттуда, потому что я думал, что они хотят взыскать налог с меня лично. Но они собирались поговорить о структуре нашей компании.

Я пошел туда, и мне сказали примерно следующее: «Чего вы хотите? Чего вы хотите от нас?» И я ответил: «Я бы хотел, чтобы вы разработали какие-нибудь стимулы, которые побуждали бы компании давать нам больше денег, потому что когда мы делаем помпу, мы устанавливаем ее на землю. Эта помпа не моя, она не принадлежит ни нашей компа-

нии, ни Anglo American» (южноафриканская шахтерская компания, которая имеет долю в PlayPumps). Она принадлежит людям. Так как сделать так, чтобы Anglo давали нам больше денег?»

Я хотел, чтобы они дали налоговые льготы. И они дали их! Они придумали такую схему: «Нужно открыть другую компанию и дать ей статус неправительственной организации». И добавили, что министрам так нравится эта идея, что нам дадут статус организации, работающей на благо общества, поскольку мы делаем именно то, что и должны делать организации подобного рода.

Мы открыли неправительственную организацию и назвали ее Roundabout PlayPumps. Мы сохранили Roundabout Outdoor, нашу изначальную компанию, потому что ее телефонные номера, факсы и электронные адреса были на всех визитках и во всех заголовках писем. Неправительственная организация принимает пожертвования, а Roundabout Outdoors назначена внедряющей организацией для этой структуры. Вот такова схема нашей работы. Но действительно бывали и невероятные проблемы.

Эндрю: **Похоже, вы прошли длинный путь. Очень вдохновляюще. Можете ли вы что-то посоветовать тем, что хочет творить добро, но несколько страшится трудностей? Как сохранять мотивацию, даже не зная, как обернется дело?**

Тревор: Ну что же, с точки зрения мотивации я придумал эту идею в 1989 году, а моя жена руководила установкой двух первых помп в Квазулу-Натале в 1993 году. В то время я не мог бросить постоянную работу – все та же старая история.

Как чего-то добиться? Следуйте лозунгу Nike: просто сделай это! Иди и делай – и все приложится. Однажды я ушел с прежнего места работы и решил, что хочу заниматься помпами все свое время.

У нас тогда было всего две помпы, и надо мной многие смеялись: «Ха-ха-ха, детская карусель накачивает воду, да-да. Ладно, все в порядке, просто продолжай принимать эти таблетки».

Теперь никто уже не смеется. Лора Буш, Билл Клинтон и Стив и Джин Кейс из Фонда Кейса со сцены мероприятия организации Клинтона объявили о 16-миллионных инвестициях в PlayPumps, которая работает в 10 странах Африки.

Итак, от двух помп в 1994 году мы дошли до 4000 помп по всей Африке, которые помогают 10 миллионам людей. Надо просто заниматься своим делом – и все будет в порядке. Так и произошло.

Рекламные плакаты: образование, здоровье, потребительская информация

Цистерны с водой играют важную социальную роль

III

Методы

У любой команды – даже если это слаженный коллектив отличных специалистов – обычно есть проблемы. И они серьезнее проблем отдельного члена команды. Очень удручает, когда вы знаете, что технически ваша команда может решить поставленную задачу, но почему-то до сих пор этого не сделала.

И здесь изменение подхода к работе может иметь существенное значение. Собственно метод и состоит в поиске наилучшего подхода, когда команда не натыкается на одни и те же проблемы.

Есть великое множество книг о программировании – отличных, хороших и не очень. Не очень хорошие книги обещают поведать «правильный способ» делать ПО. Хорошие и даже отличные книги говорят, что содержат хороший – не обязательно *единственный* – способ создания ПО. Общее у тех и других – это методы.

Методами очень легко пресытиться, потому что, похоже, на каждый элемент проекта найдется с десяток способов. Как, к примеру, описывать предполагаемое использование вашего ПО потребителями? С помощью перечней функциональности? Вариантов использования? Если да, то каких – текстовых или визуальных, на основе UML? Или взять более традиционные последовательности «стимул–реакция» (предтечи перечней функциональности и вариантов использования)? Записать их в документ или вывесить на стене? Использовать ли каталожные карточки? Стикеры? Лист Excel? То есть буквально десятки возможностей разместить одну и ту же информацию. Все зависит от того, что больше подходит вашей команде, и единственно правильного ответа, разумеется, нет.

И это всего лишь запись того, как пользователи будут общаться с вашим ПО. А ведь почти у каждого программиста свое представление об идеальной проверке кода. И о том, как документировать предложения команды по архитектуре. И о том, как фиксировать расписание проекта – один *просто ненавидит* диаграммы Гантта, другой жить без них не может.

Трудно переоценить вред, который изменения могут нанести командной культуре. И не только культуре: перемены могут поставить под сомнение само существование команды. Когда команда решается принять на вооружение новый метод, она рискует вдвойне: во-первых, сам метод может оказаться дурацким, просто тот, кто его продвигает, об этом не знает. Один из самых популярных и злополучных методов для команд – собрания. Мы не утверждаем, что все собрания бесполезны. (Любой член хорошей Scrum-команды расскажет вам, что такое по-настоящему эффективное собрание!) Но есть такие собрания, которые, можно сказать, универсально бесполезны и устраиваются главным образом для проект-менеджера или топ-менеджера. Все члены команды

усаживаются за стол, и каждый ждет свой очереди (потихоньку проверяя электронную почту). Когда очередь доходит, сотрудник рассказывает все, что сделал за прошлую неделю. Непонятно, фиксируется ли где-то эта информация, но обычно она нужна только для удовлетворения самолюбия какого-нибудь топ-менеджера. Никто не слушает других, и обычно это лишь пустая трата нескольких рабочих часов в неделю.

Подобный метод непригоден для команды, потому что никто не спросил ни у одного из членов команды, дало ли это ему хоть что-нибудь, помогло ли улучшить ПО. Если бы кто-то задал такой вопрос, собрания бы тут же прекратились.

А что если сам по себе метод хорош, но команда его просто не понимает? Здесь источник второй большой проблемы. Именно с этой проблемой сталкивались многие команды, пытаясь наладить проверку кода. Большинство программистов согласны, что проверка кода – хорошая идея. Посмотрите на самые успешные известные проекты с открытым исходным кодом – Apache, Linux, Firefox – и вы поймете, что эта культура построена на проверке кода. Но часто проверкой кода пренебрегают, потому что программисты начинают стонать и жаловаться, видя, что запланирована проверка кода. А если она все же проводится, то изменения практически незаметны, потому что в ее пользу не верят и проверяют не очень тщательно. Получается замкнутый круг: все решают, что проверка кода ничего не дает, и перестают ею заниматься. Так обычно и бывает, когда команда не убедилась в достоинствах новой методики.

Если вы хотите внедрить в своей команде новый метод работы, нужно сделать две вещи. Во-первых, надо убедить всех членов команды, что дело того стоит. Во-вторых, нужно доказать тем, кто оплачивает счета, что лишние усилия и время на эти методы – то есть, с их точки зрения, деньги – окупятся. Потому что хороший метод не сразу, но приносит пользу, а это совершенно неочевидно для многих, особенно для тех, кто сам не программирует. Попробуйте сказать своему боссу, что собираетесь применить новый метод, из-за которого разработка проекта пойдет вдвое медленнее, но зато тестирование и внедрение настолько ускорятся, что потраченное время окупится с лихвой, – и вы увидите, что мы имеем в виду.

Чтобы понять ценность нового метода, необходимо своего рода предвидение. Нужно уметь продать метод так, чтобы убедить руководство за него заплатить, а команду – его применять. И уж во всяком случае требуется команда выше среднего уровня и достаточно восприимчивая, чтобы изменить методы работы.

От избранного командой подхода сильно зависит успешность ваших проектов. И хотя правильных путей больше одного, но если команда не

придет к взаимопониманию по поводу способа создания ПО, ее участники рискуют начать работать друг против друга.

Рассказы и интервью в этой части – о методах. Но и не только: они о командах, которые ищут свой путь к наилучшим методам, подходящим именно для них, потому что для командной культуры нет ничего губительнее перехода на метод, неприменимый для данной команды. Мы отобрали эти рассказы, потому что они повествуют о важных, прогрессивных командах, которые радикально изменили свой подход к делу и добились отличных результатов. Эти руководители, видя постоянные серьезные проблемы, мешающие их командам делать лучшее ПО, нашли способ привести своих подчиненных к более совершенным методам.

15

Создание команд: сотрудничество и обучение

Джеймс Греннинг

Где-то примерно в 2000 году у меня был интересный опыт. Меня пригласили помочь одному из наших клиентов запустить новый проект. Эти люди были новичками в объектно-ориентированном проектировании, и я должен был помочь им с определением ОО-архитектуры и составлением изначального плана. Как выяснилось впоследствии, я провел с ними полгода, работая через неделю. Это стало отличным опытом для меня и для них. Мы не только добились некоторых прекрасных результатов, но и провели некоторое время в хаосе.

Компания – назовем ее BBS (By the Book Systems – Очень правильные системы) – была жестко структурирована и могла похвастаться качеством и соответствием процесса ожиданиям. Как и многие крупные компании, иногда BBS поступала себе во вред. В этом рассказе вы узнаете об отличных проектах BBS и о некоторых грубых ошибках, которые могут повредить командному духу и притормозить прогресс.

Начало нового проекта – это всегда весело. Я имел удовольствие за свою карьеру начать довольно много проектов, и влиться в команду BBS было здорово. Как консультант я часто остаюсь чужаком, работая с командой только краткие промежутки времени. Но в данном случае я работал в BBS почти каждую неделю. Так что я действительно стал одним из членов этой команды.

Вначале коллектив был небольшой: Джонни (системный инженер), Алан (ведущий инженер) и я.

Первым делом Джонни ввел меня в курс требований к проекту. У него был 30-страничный документ с требованиями (requirements), написан-

ный в полном соответствии с правилами, со всеми «должен» и «необходимо», как и полагается.

В требованиях было много полезной информации, но в такой форме она почти совершенно не годилась. Даже при определении требований было непонятно, что должна делать программа. Ведь она должна что-то делать, и нам нужен был перечень ее функций. К счастью, такие проблемы решаются просто. Варианты использования (use cases) служат для каталогизации и определения поведения системы. Это помогает понять, что программа могла бы, может и обязана делать.

Однако была небольшая проблема. Джонни до того не работал с вариантами использования. Если бы они не были частью стандартного процесса, пришлось бы искать какие-то формальные оправдания. Но мы заглянули в руководство по процессу и, к большому нашему удовлетворению, обнаружили, что варианты использования – одна из приемлемых форм определения требований. Хорошая новость!

Мы с Джонни закрылись в его кабинете. Читая требования, мы обсуждали, что значит то или иное из них и каким должно быть поведение системы. Мы перечисляли вариант за вариантом и заносили их в таблицу. После почти недельной работы мы выудили из документа Джонни 105 вариантов использования и определили девять акторов за пределами нашей системы. Мы определили границы системы и модели ее поведения, по крайней мере на уровне разговоров и планирования. Эта деятельность называется *определением вариантов использования*. Предполагалось найти все модели поведения.

Примерно в это время Кент Бек опубликовал свою книгу «Extreme Programming Explained»[1][1]. Поднялась шумиха. Тогда как раз бушевали войны по поводу только что созданных Yahoo! Groups и группы новостей comp.object. Возникли замечательные возможности. Компания Object Mentor, на которую я в то время работал, монополизировала рынок экстремального программирования (далее XP). Это не совсем правда, но звучит круто. На самом же деле Боб Мартин, Джим Ньюкерк и Лоуэлл Линдстрем (руководители Object Mentor) объединились с Кентом Беком, Роном Джеффрисом и Мартином Фаулером и стали учить людей XP. Лоуэлл пригласил меня на первое «Погружение в XP». Я взял неделю отпуска в BBS и погрузился. Эта незапланированная неделя в отеле «Мариотт» города Дирфилд (Иллинойс) имела для меня огромное значение. Удивительно, как незначительные события могут оказывать такое сильное влияние.

Надо сказать, что большинство населения мира об XP ничего не знало, и я мало отличался от этого большинства. А то, что я слышал, меня

[1] Бек К. «Экстремальное программирование». – СПб.: Питер, 2003, 224 с.

раздражало. Как может быть хорошим вариантом что-то со словом «экстремальное» в названии (кроме горнолыжного спорта)? Запасаться шлемами, налокотниками? Оформить новую медицинскую страховку? Думаю, Кент провоцировал, и я рад, что так и было. Если бы его курс назывался «Разумное программирование», все бы зевнули, и каскадный подход продолжал бы считаться идеалом. Но в итоге эта простая книжица заставила многих изменить свои взгляды и положение дел.

Проведя неделю на «Погружении», я вернулся в BBS с энтузиазмом неофита. Да, я преувеличиваю. Я никогда не творил себе из XP кумира, но считал, что некоторые его идеи могут помочь выйти из тупиков в проекте BBS.

Я начал рассказывать своим коллегам об интересных и революционных техниках в XP. Их интерес был сдержанным: «Отдел качества никогда не позволит нам заняться ничем „экстремальным“», – заметил Алан. Джонни подозревал, что требуется больше предварительной работы по проектированию и документации, чем предписывало XP. Культура длительных процессов не позволяла думать, что такое легковесное решение, как XP, может оказаться хорошей идеей.

Управление продажами

После еще нескольких разговоров об XP Алану и Джонни, наконец, понравилась идея. Мы кратко ввели в курс дела нашего менеджера Фреда, и он был полностью за, но нужно было еще сходить к нашему директору Баду. Мы знали, что тут будет ряд спорных вопросов, особенно по безопасности процесса и поддержанию надлежащего уровня документации. Документационные требования в BBS были строгими. Но моим долгом как тренера команды было изменить их текущие методы и помочь им в росте.

Я подготовил презентацию по методам XP. Мы предугадали некоторые вопросы, которые должны были возникнуть, и создали вариант для наших рекомендаций. Презентация заканчивалась несколькими открытыми вопросами, которые должны были инициировать диалог.

Бад: «Все это звучит прекрасно, но как быть с техническими проверками, которые предписывает процесс?»

«Нам нужно отказаться от проверок как от критического пути», – ответил я Баду. Я не сказал ему ничего нового, лишь то, что проверки временно задерживают прогресс. Подготовка к проверкам занимает много времени, к тому же всегда было технически сложно вписать наши собрания по проверке в расписание старших инженеров. Мы обсудили, как текущий подход влияет на неопробованные идеи по проекту.

Я объяснил: «Самые опытные ваши люди большинство времени тратят на проверки, а не на создание продукта. Каждой команде нужен опытный проектировщик». Мы поговорили о том, что проектирование – это ежедневная деятельность, а не нечто одномоментное.

Бад: «Думаю, я понял. Вы говорите, что нам не нужны проверки?»

«Мне не нравится идея отмены проверок: мы изменим то, что и когда нужно проверять», – ответил я. Мы понимали, что периодических проверок не избежать, но можно было найти для них значительно более эффективный способ. Проверки тормозят дальнейшую работу команды, пока собрания по проверке продолжаются, а проектированием пренебрегают. Мы сказали Баду, что, пожертвовав проверками как критическим путем, мы можем сохранить темп работы команды, которая будет во время них заниматься проектированием, реализацией и тестированием.

«В настоящее время самые опытные ваши сотрудники, вместо того чтобы создавать проект, проверяют его», – сказал я. Далее мы обсудили то, как всего один опытный проектировщик в команде может с помощью ежедневных обсуждений проектирования и программирования в парах поддержать качество проекта на высоком уровне.

Бад спросил: «Хорошо, ну а что тогда проверять?»

Методы экстремального программирования

Экстремальное программирование – это набор принципов и методов для постоянного создания высококачественных программ. Это один из первых членов семейства методологии Agile-разработки.

Потребитель – член команды

Команды включают в себя человека (или группу людей), представляющего интересы потребителя. *Потребитель* решает, что должно быть в продукте и чего не должно быть. *Потребитель* отвечает за приемку продукта. Это означает, что команда потребителей, по всей видимости, должна состоять из опытных тестеров.

Перечень функциональности

Перечень функциональности представляет функции системы. Эти перечни коротки настолько, чтобы один человек мог исполнить их за одну итерацию. Если перечень слишком длинен, чтобы его можно было исполнить за одну итерацию, его нужно разбить на более мелкие. Перечень – то же самое, что вариант использования.

Планирование

На этом этапе потребитель и программисты определяют цели новой программы. Программисты оценивают усилия, которые нужно приложить для исполнения каждого перечня. Потребители выбирают перечни и оформляют их в итерации, которые не превосходят возможностей команды. Перечни заказываются потребителем в соответствии с их значимостью и ценой.

Небольшие партии

Система создается небольшими партиями. Каждая итерация длится две недели. Партия – это группа итераций, которая обеспечивает важные функции пользователям этой системы.

Приемо-сдаточное тестирование

Приемо-сдаточные тесты демонстрируют, что перечень окончен. Они показывают детали каждого перечня и определяются еще до его запуска. Приемо-сдаточное тестирование выполняется потребителями. Большинство тестов может быть автоматизировано, так что их можно запускать в любое время, нередко по несколько раз в день.

Открытая рабочая среда

Чтобы облегчить рабочее общение, команда работает в открытой рабочей среде, где облегчен доступ к другим членам команды, оборудованию и информации о статусе проекта.

Проектирование по результатам тестирования

Программисты пишут программы очень маленькими шагами, постоянно их проверяя. Сначала пишется небольшой тест, за которым следует ровно столько кода, чтобы удовлетворить его условиям. Затем пишется другой тест и так далее.

Метафора

Системная метафора дает системе идею или модель. Это часть высокоуровневого проектирования, облегчающая общение. Она обеспечивает контекст для именования модулей, классов и функций программы.

Рефакторинг кода

Рефакторинг кода – это процесс сохранения чистоты проекта. Когда система развивается, проект может загрязняться. Цель рефакторинга состоит в обнаружении, определении и устранении проблем проектирования по мере их возникновения, в случае если эти проблемы невелики и могут легко быть устранены.

«Мы будем проверять то, что уже сделано и протестировано, а не то, что предполагается сделать», – ответил я Баду. И рассказал, что во время каждой итерации мы запишем важные архитектурные решения и идеи. Если во время проверок выявятся ошибки проектирования, можно будет с уверенностью вносить изменения. Наши автоматизированные тесты отметят поведение системы. Любые необходимые изменения можно будет внести при следующей итерации.

«Давайте уточним значение документа по проектированию. Его цель – обеспечить высококачественную дорожную карту системы, – добавил я. – Детали должны быть в коде и в тестах».

Бад: «Секунду, так детальной документации не будет?»

«Вот именно, – ответил я. – Программируя с предварительным тестированием, мы будем осуществлять множество автоматизированных модульных тестов. Тест-кейсы и будут нашей детальной документацией». И я рассказал, что каждый тест-кейс – это фокусный пример кода с определенными предварительными условиями и желаемым результатом.

Я добавил: «Тесты – это исполняемая спецификация». Как только мы научимся понимать результаты тестов, они станут очень эффективной формой детальной документации. Это не те прозаические документы, которые нужно прочесть и как-то интерпретировать. Эту спецификацию можно исполнить и рассмотреть на предмет любых отклонений, как сейчас, так и в будущем. «Это как неразменный доллар», – отметил я.

Все еще не очень уверенный в успехе, я добавил: «Бад, вы знаете, что мы в этом деле новички, но я думаю, это имеет смысл. И в любом случае у нас же месячные циклы работы: мы можем попробовать этот подход и каждый месяц пересматривать решение».

Бад: «Хорошо, так и сделаем». Но тут он спросил: «А как узнать, что тесты верны?»

«Мы уже об этом думали, – я помолчал и продолжил: – Мы считаем, что здесь нам поможет программирование в парах, но, наверное, тест-кейсы нужно проверять. Мы хотим убедиться, что модуль работает правильно, а его интерфейс удобен в использовании и хорошо спроектирован».

Бад: «Хорошо, мы будем проверять тесты, чтобы понять, верен ли интерфейс и то ли, что нам нужно, тестируется».

«Да, в этом и состоит идея». Мы перешли к дальнейшей дискуссии и в итоге согласились, что принимаем верное решение. Мы сошлись на том, что определенные детали реализации могут быть не так важны, как хороший интерфейс и правильное поведение системы. Последние факторы сильно влияют на общую архитектуру. Я добавил: «Не забывайте,

что если реализация будет нуждаться в улучшении, то у нас будут тесты, которые позволят убедиться, что после рефакторинга сохраняется желаемое поведение системы».

Бад: «Все это хорошо, но наши процессуальные требования предписывают проверки кода, и этот метод всегда нам помогал. Что теперь будет с ними? Их место займет программирование в парах?»

«Мы так думаем. Собственно, это и есть одно из тех *экстремальных* решений, которые нас несколько беспокоят, – добавил я. – Проверки кода при программировании в парах происходят постоянно». Я вкратце пояснил принципы работы программирования в парах. Мы проверяем друг друга по очереди, может быть, раз в несколько часов. Это своего рода проверка кода вживую, но во многих отношениях она лучше, хотя в некоторых других может быть хуже. Два человека принимают одни решения по проектированию, дают имена классам, функциям и переменным в коде. Негативная сторона в том, что ошибаться могут оба.

Бад знал, что в проверки кода вкладывается много денег. Много времени занимали подготовка и собрания. Мы решили, что программирование в парах поможет сэкономить время и средства. Ему понравилось то, что он услышал. Все мы постановили, что сочетание программирования в парах с проверкой тест-кейсов приведет к лучшим результатам, нежели текущий процесс, связанный с множеством документов.

Бад: «Мы будем проверять код в реальном времени, чтобы убедиться, что наши стандарты выдерживаются. Тест-кейсы покажут, что должен уметь код, и подтвердят, что он это умеет».

Бад был крут. Он понимал, что текущий процесс тормозит развитие команды, и верил, что основанные на здравом смысле методы XP будут способствовать нашему прогрессу. Он понимал, что нужны изменения. Ему очень понравилось потенциальное улучшение предсказуемости расписания, качества и удовольствия от работы. Он сказал, что считал бы XP-проект успешным, даже если бы выполнялось одно условие из трех. Бад дал нам зеленый свет для перехода на экстремальное программирование. Мы получили также карт-бланш для защиты от контролеров процесса: «Я подпишу любые документы, которые вам могут понадобиться», – сказал Бад.

Сейчас я могу сказать: Бад дал мне понять, что непременным условием для команды является хороший менеджер, обладающий смелостью, предвидением и желанием поддержать свою команду. Бад поддержал нас, доверился нам, и это помогло нам спокойно выйти за рамки установленных норм.

Как начиналось

Мы могли набрать команду сразу, но решили подождать месяц, пока у нас не будет первой итерации. Таким образом, мы начали свой процесс познания и приобретения опыта. После одной-двух итераций мы могли обучать новичков, меньше опираясь на догадки

Джонни – теперь наш *потребитель*, – Алан и я выписали все варианты использования на карточки. Мы хотели, чтобы они были удобны в применении, точь-в-точь как меня учили на «Погружении» Кент и Рон. Там же, на «Погружении», я узнал и о *перечнях функциональности*. Мне казалось, что перечень функциональности и вариант использования – практически одно и то же. Мы собрались было перейти с вариантов использования на перечни функциональности, но тут Джонни предупредил меня: «Если мы собираемся пользоваться перечнями, то нужно будет писать объяснительную записку о том, почему мы отклонились от стандартного процесса». Джонни также заботил неформальный характер перечней функциональности, и он добавил: «Кроме того, мне кажется, что перечни функциональности слишком бедны деталями. Этого никогда не одобрят».

Мы решили продолжать использовать варианты: в конце концов, этот метод в BBS был принят. Мы считали, что нам предстоит расширить горизонты во множестве отраслей, и варианты казались для этого подходящим инструментом. Мы решили, что не надо сразу же вступать в борьбу, а свои карт-бланши использовать только тогда, когда они действительно понадобятся.

Перед самым началом первой итерации мы с Джонни выработали детальные шаги ключевых вариантов. Ключевые варианты имели наибольшее значение и должны были оказать самое сильное влияние на общую архитектуру. Теперь мы были готовы к итерации.

Забегу немного вперед: после первой итерации мы стали подумывать о методе автоматических *приемо-сдаточных тестов* для XP. Как и предлагали на «Погружении» Кент и Рон, мы сделали специальный тестовый язык для приложений. Мы хотели, чтобы язык получился читабельным и не слишком заточенным под программистов, с той целью, чтобы любой компьютерный специалист, не будучи программистом, мог читать и писать тесты. Через несколько итераций мы поняли, что информация в вариантах и информация в приемо-сдаточных тестах во многом дублируют друг друга. Дублирование же – как раз одна из тех помех, которые призвано предотвратить XP.

Однажды я написал сценарий, который вовсе не определял варианты использования. Вместо этого мы писали приемо-сдаточные тесты до

итерации, указывая детали вариантов использования. Эти тесты были исполняемой спецификацией! И так мы перестали тратить усилия на создание вариантов использования, которые Джонни должен был писать одновременно с приемо-сдаточными тестами.

Как оказалось, для BBS такое решение было слишком уж экстремальным. Пришлось заниматься и вариантами, и тестами, поскольку это было проще, чем еще раз расшатывать систему. Полагаю, Джерри Вейнберг назвал бы это соленым огурцом: если поместить рядом свежие и соленые огурцы, то что будет? «Свежие огурцы станут похожи на соленые больше, чем соленые на свежие» [3]. Итак, наше XP немного просолилось, как и я.

Вернемся к началу нашей первой итерации. Оценив несколько вариантов, Джонни запустил ее. Изначальные варианты описывали ключевые функции, которые охватывали самые частые операции: система должна была выполнять их тысячи, если не миллионы раз на дню. Мы с Аланом приступили к работе. К нам время от времени присоединялась Эллен, которая работала с нами неполный рабочий день. Она была старшим инженером, отдыхавшим от последнего проекта, и была готова начать с нами экстремальное программирование. Мы договорились, что весь код будет написан в парах. Очень хорошо, что у нас оказался третий член команды, потому что пока Алан ходил на свои собрания, я мог работать с Эллен, и наоборот.

Первая итерация оказалась успешной. Мы сделали большинство из того, что запланировали. Сейчас я могу сказать, что это было серьезное достижение. В Object Mentor мы относились к первой итерации как к «нулевой». И не потому, что гики любят начинать отсчет от нуля, а просто слишком много первых итераций закончились ничем. Это был наш первый XP-проект. Мы не знали, что должны провалиться, и вместо этого разобрались с основной функциональностью.

Мы не сражались за каждый метод XP. Как я впоследствии убедился уже в качестве тренера, желание команд учиться и пробовать новое обеспечивает успех проекту. Да, книгу написал Кен Бек, но это был наш процесс.

Мы изучили азы проектирования с предварительным тестированием и постоянной разработкой. Наш менеджер Фред был в восторге от результатов: «Раньше за месяц мне представляли только черновую спецификацию требований, а сейчас у нас работающий код!»

Итак, базовые знания и начальный код у нас были. Пора было увеличить команду.

Рост команды

Теперь, когда Алан, Эллен и я уже разбирались в новых методах, мы пригласили еще пару новых смышленых инженеров – Пола и Алекса. Им было интересно попасть в команду, где учатся ОО-проектированию и XP, и внести свою лепту в новый продукт.

Это была хорошая команда, нам отлично работалось вместе. Мы сотрудничали и радовались прогрессу и познанию. Мы программировали в парах, писали CppUnit-тесты, перерабатывали проект с продвижением вперед и создавали собственный язык для приложений, чтобы проводить приемо-сдаточные тесты. Мы добились многого.

Расширение горизонтов и контролеры процесса

Контролеры процесса во главе с Мэрили сидели у нас на хвосте из-за легковесности нашего подхода к документации и отклонении от каскадного подхода с его постепенным процессом. Только карт-бланш от Бада нас в какой-то мере спасал. Но нарушать нормы приходилось довольно регулярно.

К нашему первому собранию по проверке мы закончили несколько вариантов использования. Проект был простой, так что проста была и документация. Она заняла только часть дня. Соответственно, документов было мало. У нас было двое проверяющих, Джим и Арт. Представьте себя в роли контролеров процесса. Они просто не привыкли к таким тоненьким пачкам документации. Но постойте! Ведь это информация по работе всего за месяц.

То, что у нас было благословение Бада на изменение статус-кво, решило дело. Очевидно, наши решения по проектированию оказались достаточно удачными, потому что большую часть времени мы обсуждали процессуальные вопросы и то, как нам удастся выкрутиться с такой тонкой пачкой документов.

Для следующей проверки мы приготовили несколько больше бумаг, но им требовалось еще больше. Хотя никто не мог сказать зачем. В итоге мы решили продолжать писать нашу тонкую архитектурную документацию, хорошо переработанный код, понятные сегменты и приемо-сдаточные тесты.

Обучение

Пол и Алекс были относительно новыми сотрудниками BBS – работали всего один или два года, если я ничего не путаю. Большую часть времени пришлось проводить с ними. Старшие должны были по-прежнему

ходить на собрания, так что мы втроем могли работать сколько угодно. Казалось, что у меня, сезонного наемного рабочего, появилось двое подмастерьев. Мы работали, день ото дня занимаясь разработкой. Как я уже говорил, я работал в BBS в основном только через неделю. Так что по пятницам нужно было планировать действия на всю следующую неделю.

Пол и Алекс были отличные ребята. Они всегда делали то, что планировалось. Код был хорошо протестирован и обладал необходимой функциональностью. Как руководитель я проверял их работу по понедельникам. Код у них всегда работал, но некоторые его сегменты были помещены не туда. То, что я обнаруживал у них, Мартин Фаулер определяет как «функциональная зависть» [4].

Теперь у проблем с кодом есть названия! Фаулер и Бек отлично поработали, дав названия и описав некоторые «признаки плохого кода». Функциональная зависть – это случай, когда один класс делает работу, которую должен делать другой класс. Есть и другие ошибки: например длинный метод, большой класс, длинный список параметров, ленивый класс, хирургия при помощи дробовика. Уже то, что мы знали названия некоторых нежелательных качеств кода, помогло нам обнаружить проблемы на ранней стадии и очистить код.

Итак, после Алекса и Пола оставались небольшие проблемы с кодом, но ничего страшного: понедельник все равно был днем рефакторинга. У нас были тесты, так что завистливый код можно было просто и без риска перенести в соответствующее место. Это был полезный опыт. Я бы никогда не додумался назвать это *функциональной завистью* без знания XP. Пол и Алекс могли испытывать чувство вины из-за этих ошибок, но все же дух познания был сильнее. Они тоже многому меня научили за время совместной работы.

Через некоторое время это превратилось в игру: Пол и Алекс заканчивали работу и начинали охоту за ошибками. Утром в понедельник я приходил, наливал себе кофе и искал их. Пол обычно говорил: «Джеймс, мы провели все тесты, но тебе наверняка кое-что не понравится». Они уже учились успешному рефакторингу! А для этого нужно было самим научиться находить и определять признаки плохого кода. Они не всегда знали, как с ними бороться, но хотя бы понимали, как выявить проблему. Осознание ошибок – первый шаг к изменениям.

Парное программирование очень помогло прорвать коммуникационные барьеры. Когда несколько часов сидишь бок о бок с другим человеком, проявляются твои собственные сильные и слабые стороны. И это улица с двусторонним движением. Если люди и отношение к делу соответствуют, то возрастает доверие и появляется понимание. Атмосфера в нашей компании давала право на ошибку и возможность учиться.

Требования для локальных потребителей

Джонни был специалистом по системам. Он создавал и программировал исходную систему за несколько лет до того. Когда я работал над проектом, он был системным инженером. Благодаря своим обширным знаниям о системе он писал спецификации. Мы вместе извлекали имена вариантов, соответствующих требованиям. Джонни вырабатывал требования как раз вовремя, чтобы мы могли спроектировать и применить их. Джонни всегда должен был торопиться, чтобы выдать следующую порцию вариантов. Иногда у него занимало больше времени выработать варианты, чем у нас – применить их. Это его всегда огорчало. Дела шли очень хорошо. Я начал ценить то, как важно иметь локального потребителя в команде.

Как-то раз мы с Джонни и Полом закончили проверку вариантов и собрались приступить к разработке. Мы рассчитали влияние нового варианта на имеющийся проект. При этом мы пользовались высокотехнологичным устройством, известным как классная доска. Проект системы был у нас в голове, так что мы перерисовывали проект на доску и обсуждали изменения. Когда мы удовлетворились текущим ходом дел, Пол и я пошли за его стол, чтобы начать.

Через несколько минут возникла проблема. Мы еще раз прочитали детализированный вариант. Его можно было интерпретировать поразному. Мы буквально полчаса назад обсуждали это с Джонни, но Пол сказал: «Джонни говорил, что нам нужно попробовать заново провести транзакцию в этих условиях». Я ответил: «Нет, в этом случае нужно записать сообщение и продолжить. А ты говоришь о варианте с расширением 4B, им мы будем заниматься через неделю».

Конечно, мы обсуждали это (собственно говоря, спорили) еще минут 20. Потом на меня снизошло: ведь нет надобности строить предположения, у нас есть локальный потребитель! «Давай спросим Джонни», – я качнул головой в сторону стены. Джонни был в своем кабинете в пяти метрах отсюда. Странно, что он нас не слышал. Джонни решил вопрос меньше чем за минуту без всякой электронной почты, голосовых сообщений и циклов проверки требований. Мы спросили Джонни, он разобрался, и мы вернулись к работе.

Кто-то может сказать, что нам нужна была лучшая спецификация, кто-то добавил бы, что варианты следовало детализировать. Было бы здорово действительно иметь эти документы, но вряд ли это нам сильно помогло бы или оправдало затраты. Документ никогда не может стать настолько полезным и детализированным, чтобы не пришлось обращаться за разъяснениями к потребителю. Единственное, что содержит все детали, – это код.

Были и другие случаи, когда нужно было прояснять требования. Много раз Джонни просился отойти и изучить базу имеющегося кода. Унаследованное приложение, которое мы сейчас перерабатывали, фактически и было нашим документом о требованиях. Это был единственный документ, детализированный настолько, чтобы ответить на некоторые тонкие вопросы. Если бы все эти вопросы были включены в документ, то он был таким же детализированным, как код, и при этом вряд ли очень полезным.

Проблема в Ривер-Сити

Все шло прекрасно до одного рокового утра понедельника. Уолли, наш новый инженер, пришел неожиданно в очень агрессивном настроении. Он что-то не поделил с Эллен и сразу же начал отпускать грубые шутки, которые ее ранили. После того как я велел ему замолчать, он заявил, что просто дурачится, но было видно, что ей больно. Острые как бритва шуточки продолжались.

В больших компаниях часто имеют дурную привычку обращаться с людьми как с винтиками в механизме. Первый признак этого – именование людей *ресурсами*.. Кто-то в руководстве подумал, что Уолли – это заменяемый винтик. И из-за этой идеи о «восполняемости ресурсов» и приортитетности проекта Уолли и появился в то роковое утро у нас.

Уолли был умен, но совершенно не понимал нашей командной культуры. Нашей команде нравилось работать вместе, и мы были рады возможности принять вызов, брошенный нам техническими и процессуальными сложностями проекта. Все члены команды ассоциировали себя с коллективом и хотели быть его частью, живя по тем стандартам трудовой этики, которые мы приняли. Уолли не понял или не захотел понять, зачем мы пишем тесты. Он не понимал, почему бы сначала не сделать проектирование. Ему не нравились краткие итерации, и к тому же он оказался неудачным партнером по программированию. Его *юмор* дезорганизовывал как на собраниях, так и во время парной работы.

Я сразу же понял, что Уолли нам не подойдет. Фред хотел, чтобы мы попробовали внедрить его в свою команду. Мы пытались, но с тем же успехом можно было чистить песком контактные линзы. В конце концов Фред принял нашу сторону и вывел его из команды. Но за то короткое время, что он был с нами, наши боевой дух и производительность труда упали.

Команды состоят из людей, а не ресурсов

Из этого следует важный вывод. Люди – это не заменяемые винтики. Команда может рассыпаться из-за одного члена коллектива. Особенно

это заметно в корпоративной команде. Предполагаю, что людей типа Уолли надо прятать по отсекам и поручать им индивидуальные задания. В них они могут быть даже вполне успешными. Но это не было характерно для нашей команды, мы не привыкли работать таким образом. В моей карьере самыми приятными были периоды такого сотрудничества, как в этот раз. Наименее удачно складывалась работа из-за плохих взаимоотношений.

Когда Уолли, наконец, ушел, все вернулось на круги своя. Работать было весело, и мы многое сделали.

У каждой компании свои проблемы

Один из методов XP состоит в принятии на вооружение открытой рабочей среды. То есть у команды есть собственное рабочее пространство. Посередине стоит большой стол. Информация о проекте прикреплена к стенам. Компьютеры установлены на столе оптимальным образом для совместной работы.

У нас открытой рабочей среды не было, и мы поняли, что работать в отсеках неудобно. Мониторы стояли по углам, отчего работать вместе было трудно. Мы стали поговаривать о том, чтобы снести стенки некоторых отсеков, чтобы освободить место для общения и сотрудничества.

Я попросил Джонни убрать стенки. Он ответил: «Нельзя, у нас есть специальные дизайнеры мебели. Надо идти сначала к ним». В моем воображении возник Питер Гиббонс, программист из фильма «Офисное пространство», который наклонил стенку, и она упала. У нас это было невозможно. Надо было навестить Бада.

Я сказал: «Бад, стены отсеков вредят нашему общению. Нужно убрать несколько отсеков и освободить пространство для команды».

«Прибереги свою энергию для чего-нибудь другого, а этого нельзя. Нам запрещено передвигать мебель, – вздохнул Бад. – Как только стена или рабочее место передвигаются, – тут он указал в направлении 50 или даже большего количества отсеков, – все пространство должно быть приведено в соответствие с текущими спецификациями».

Я ответил: «Так, и что это значит? Можно нам сделать открытую рабочую среду?»

Бад: «Здесь я не уверен, но если передвинуть одну стену, – ответил он, указывая на личные кабинеты вдоль внешней стены, – то все старшие инженеры и менеджеры лишатся личных кабинетов и дверей. Этого же никто не допустит».

Бад утешил нас тем, что если это входит в наши планы, то он постарается что-то сделать: «Но передвижение стены коснется всего коллектива, это неизбежно».

Битва за стену закончилась, не начавшись. Можно было сколько угодно дуться и скулить. Вместо этого мы вернулись к работе и стали жить с тем, что было до этого. Что ж, мы еще немного просолились.

Следующие проекты

После успеха первого проекта Фред предложил еще нескольким командам заняться *экстремальным программированием*. Но это оказалось труднее ожидаемого. Во-первых, Фред хотел, чтобы каждая новая команда управлялась так же, как и наша первая. «Так мы делаем пилотные проекты», – сказал мне Фред. Разумеется, в BBS существовали правила даже относительно того, как опробовать новое, и нам не удалось избавиться от этого!

Я предупредил его, чтобы он не заставлял каждую команду применять все те же методы. Я считал, что это не сработает. Каждая команда уникальна, у каждого ее члена есть уникальные навыки. Различия неизбежны, и они заведомо будут работать не так, как мы.

Так или иначе, мы попробовали. Проекты были успешными, в результате получалось в два или три раза больше высококачественного программного обеспечения на одного среднестатистического разработчика, чем при стандартном проекте разработки BBS. Но с течением времени становилось все труднее работать одними и теми же методами. Разумеется, ключевые действия были одними и теми же, но появилось много вариантов. Каждая команда нашла собственный путь.

Одним из самых крупных препятствий, с которыми я столкнулся, была культура. В их культуру входила гордость за процессы, которые они разработали и которым следовали. Эти процессы помогли компании многого достичь. Но нужно было двигаться быстрее. Что-то нужно было изменить, но культура сопротивлялась изменениям. Через год я поговорил с ведущим инженером соседней группы и спросил его, занимается ли его команда XP. Он ответил, что это привело бы к слишком большим проблемам. Команда, которая хотела заниматься экстремальным программированием, должна была написать 30-страничное обоснование процесса, утвержденное контролерами процесса. Понятно, что не так много команд проектировщиков захотели работать в следующем XP-процессе.

Факторы, влияющие на успех сотрудничества

Мы продемонстрировали, что можно улучшить существующее положение дел. Мы бросили вызов принятым методам и обрели новое понимание того, как можно работать. Это в очередной раз доказывает, как важна хорошая команда. Работая вместе, хорошая команда способна достичь гораздо большего, чем несколько индивидов. Как же добиться спаянной, тесно сотрудничающей команды?

В блоге Алистера Кокберна [5] я нашел некоторые интересные наблюдения на тему командной работы и сотрудничества. Алистер писал: «Люди сотрудничают, когда хотят, – и продолжал: – А когда они хотят?» В неформальном исследовании он обнаружил, что определенные действия приводят к атмосфере тесного сотрудничества. Вот вкратце некоторые из них.

- Воодушевляй других: цени других.
- Увеличивай уверенность: поддерживай других, спорь, но перенимай идеи.
- Прогрессируй: успех рождает успех.
- Больше энергии: бросай вызов, вноси свой вклад.

В этой истории описанные Алистером действия проявились сполна.

Вера Бада в нашу команду и готовность помочь нам действительно очень нас поддержала. Наши стили обучения и преподавания поощряли людей. Антипримером было назначение в команду Уолли. Он не подбадривал людей, а расстраивал.

Карт-бланши Бада и его одобрение наших действий дали нам возможность пробовать новое. Его реалистические ожидания добавляли нам ощущения уверенности.

У нас в команде были отличные ребята: квалифицированные, но при этом не считающие, что знают все. Оптимистичный подход команды и постоянная готовность к обучению сделали работу удовольствием. Ранний успех внушил уверенность в своих силах и дал отличный толчок делу. Мы обнаружили, что все работает, что прогресс очевиден. Каждый день случались новые достижения, связанные с сотрудничеством команды. Мы бросали вызов друг другу и организации. Это был незабываемый опыт. Конечно, мы не смогли превратить соленые огурцы в свежие, но зато изменили вкус рассола!

Литература

Beck, Kent. 1999. «Extreme Programming Explained». Reading, MA: Addison-Wesley.[1]

Grenning, James. 2002. «Extreme Programming and Embedded Software Development». Embedded Systems Conference (San Jose, CA).

Weinberg, Jerry. 1985. «The Secrets of Consulting». New York, NY: Dorset House Publishing.

Fowler, Martin. 1999. «Refactoring». Reading, MA: Addison-Wesley.[2]

Cockburn, Alistair. 2008. «Collaboration: the dance of contribution»; *http://alistair.cockburn.us/Collaboration%3a+the+dance+of+contribution*.

[1] Бек Кент «Экстремальное программирование: разработка через тестирование». – СПб.: Питер, 2003.

[2] Фаулер Мартин «Рефакторинг: улучшение существующего кода». – СПб.: Символ-Плюс, 2008.

16

Эффективные методы

Интервью со Стивом Макконнеллом

С самого начала нашей работы в ИТ мы находились под влиянием Стива Макконнелла. Он был одним из первых прочитанных нами авторов, которым удалось написать простым и понятным языком о сложных вопросах проектирования и обеспечения качества ПО. Стив во многом определил и стиль наших книг в выборе как тем, так и выразительных средств. В каком-то смысле именно Стив показал нам, что можно писать серьезные книги так, чтобы они выглядели как непринужденный разговор.

Еще одним человеком, очень повлиявшим на нашу жизнь, стал Скотт Беркун; именно благодаря ему мы сегодня пишем книги для O'Reilly. Скотт пригласил Эндрю в качестве технического эксперта при написании своей первой (и очень хорошей) книги «Making things happen» (Добиваясь результатов), и именно тогда зародились наши отношения с O'Reilly. (Скотт, в свою очередь, оказал нам неоценимую экспертную поддержку при написании книги «Applied Software Project Management» (Прикладной менеджмент проектов разработки ПО). Поэтому, когда он предложил нам взять интервью у Стива для этой книги, мы с радостью схватились за эту возможность свести вместе двух этих безмерно уважаемых нами людей и послушать их разговор о технических командах.

Скотт: **Первое, о чем бы я хотел спросить, — это говорит ли тебе о чем-нибудь выражение «идеальная команда»? Ты когда-нибудь прежде его слышал или называл так команды программистов, в которых работал или которыми руководил?**

Стив: Вообще, я все понимаю буквально, поэтому определения «идеальная», «прекрасная» в отношении команды мне не очень понятны. Когда я слышу «прекрасная команда», я полагаю, что речь идет о чем-то другом, о высокой эффективности или высочайшем качестве работы команды, как-то так.

Скотт: **Или о команде топ-моделей.**

Стив: *(Смеется)* Да, в программировании такие нечасто встречаются. То есть само это словосочетание не вызывает никаких прямых ассоциаций. Ну, может быть, мне представляется высокопрофессиональная команда с отлично выстроенными взаимоотношениями, где каждый готов помочь другому и притом чувствует свою принадлежность к группе избранных, где люди приходят в итоге к успешным результатам.

По-моему, недостаточно того, чтобы сама команда чувствовала, что работает над чем-то особенным. В какой бы сфере бизнеса они ни работали, нужно чтобы вся эта сфера или рынок признали исключительность их продукта или услуги, только тогда команду можно будет назвать идеальной.

Скотт: **Думая об эффективных, идеальных командах — назвать можно как угодно, — я всегда задаюсь вопросом, идет ли речь о процессе или о результате. Безупречен ли только рабочий процесс или, напротив, только результат их работы?**

Стив: Я почему-то снова думаю тут о качестве и, в частности, о том, как его определяет Деминг[1], то есть как о соответствии требованиям. Поэтому, пытаясь как-то представить идеальную команду, я логически прихожу к выводу: идеальная команда – это команда, которая соответствует требованиям.

А требования к командам могут радикально меняться, в зависимости от контекста. В одних случаях может требоваться высокий уровень инноваций. В других условиях может понадобиться высокая эффективность работы. А могут быть ситуации, когда требуется невероятная концентрация.

Одним из подразумеваемых требований, которым часто пренебрегают, является устойчивость, прочность команды, то есть эти люди делают

[1] Уильям Эдвардс Деминг – известный американский ученый, статистик и консультант по теории управления качеством. – *Прим. перев.*

что-то великое, но сразу же выдыхаются и как команда уже не способны на повторение такого успеха. Хорошо это или нет, зависит от организационных требований.

Конечно, бывают случаи, когда компании необходимо в краткие сроки создать крутой продукт, и если команда после этого разваливается, это можно считать ценой данного единовременного успеха. Но есть и другие организации, которые работают в более устойчивом режиме, и для таких компаний загонять и разваливать свои команды – значит ставить под угрозу свое будущее.

Скотт: **Какую из команд, в которых работал, ты мог бы назвать лучшей и почему?**

Стив: Одна из лучших команд, в которых я работал, номинально даже не была командой. Скорее, это была рабочая группа, а командой я не могу ее назвать потому, что мы на самом деле не работали над единой общей задачей. Мы работали в одной компании, но в разных проектах и для разных заказчиков.

Но атмосфера была потрясающая. Нам так хорошо работалось вместе – тут я не знаю, можно ли назвать это «идеалом» команды – потому что мы все были молоды и примерно в равных условиях. У нас было схожее чувство юмора. Никто не был связан семьями и бытовыми проблемами.

Поэтому мы проводили много времени вместе: на работе и после работы. У нас был общий враг, наш начальник, и мы чувствовали, что делаем огромную работу. То есть было чувство, что мы делаем много не благодаря руководству, а вопреки ему. Были ли мы командой? Трудно сказать. Мы были сплоченным коллективом, и цели у нас, в конечном итоге, были общими.

Скотт: **То есть фактически вы не работали вместе над одним проектом?**

Стив: Мы делали разные проекты для разных заказчиков, но находились при этом в одном помещении. И наша работа во многом пересекалась. Мы менялись и переключались с проекта на проект, поэтому между небольшими проектами, над которыми мы работали, установился очень хороший уровень сотрудничества. Ну, тут трудно судить, сразу впадаешь в ностальгию по тем добрым старым временам.

Скотт: **Ничего, ностальгия здесь вполне уместна. То есть очень немногие из тех, кто будет читать эту книгу, отдают себе отчет в том, что в настоящее время они работают в лучшей из команд и больше такого не повторится. Мы всегда думаем о лучшем опыте как о чем-то из нашего прошлого, и даже если нынешняя команда окажется лучшей**

за всю нашу жизнь, полностью мы сможем это осознать только после ухода из нее.

Стив: Согласен, и это вполне соответствует тому, о чем я сейчас говорил. Большую часть времени, что я провел с этими людьми, я скучал по своей студенческой компании. Потребовался, пожалуй, год, чтобы однажды проснуться и понять, что у меня, вообще-то, все лучше некуда.

Не понимаю, если честно, почему я устроен так, что мало думаю о своей текущей работе, чаще возвращаясь мыслями к прошлой, но в том случае именно так все и было.

До этого я работал над очень интересным проектом в университете.

Скотт: **В чем заключалась цель того проекта?**

Стив: Ну, кстати, там у нас была очень хорошая команда еще и потому, что мы четко понимали разницу между целями университета и нашими личными. Целью университета была разработка программы для заказчика. А нашей личной целью было вовремя получить диплом.

И внутри команды мы всегда очень хорошо чувствовали эту разницу между тем, как видело цель наше учебное заведение, и той целью, которую мы видели перед собой.

Скотт: **Очень интересно. Идеальная команда, ведущая подрывную деятельность.**

Стив: Мы очень активно выстраивали проект таким образом, чтобы достичь нашей истинной цели, то есть получить диплом в должный срок; в той команде было забавно то, что в начале года, когда мы только начинали работу, большинство из нас не были знакомы друг с другом.

И очень долго мы занимались тем, что пытались почувствовать друг друга, понять, кто в чем силен, кто чем хочет заниматься, у кого какие слабости и так далее.

В то время мне страшно не нравилась эта медлительность, но когда проект был уже запущен, я понял, что это было очень полезно для нас. Каждый был готов идти на уступки в случаях, когда вся команда решала, что это необходимо.

Поэтому все работали над тем, что было интересно лично им, и не получали чрезмерной нагрузки. В результате команда была очень гибкой. У нас, например, был парень, который был женат и ждал ребенка. А потом они этого ребенка потеряли. Другой человек разводился. Были двое, которые много путешествовали. Но даже при всех серьезных проблемах команда оставалась крепкой и продолжала работать. В случае необходимости каждый готов был взять что-то на себя.

Еще более замечательно то, что это не было нашей основной работой.

У нас была и другая, настоящая работа, которой также необходимо было уделять внимание. В моем понимании это была группа людей, действовавшая именно как команда, причем на самом высоком уровне. Общее целое превосходило сумму слагаемых по многим показателям.

Скотт: **Вопрос не вполне корректный, потому что я попрошу дать точную оценку абстрактным вещам, но все-таки спрошу. Сколько команд, если оценивать в процентном соотношении, можно назвать высокопродуктивными?**

Стив: По-моему, по-настоящему эффективно работающих команд очень мало.

Скотт: **Сколько? 10%? 5%? 1%?**

Steve: Если говорить о тех сотнях команд, с которыми мы работали за эти годы, я бы сказал, что меньше 10%. То есть некоторые выпадают по определению. Говоря о высокой эффективности, вы подразумеваете, что команда выходит за рамки среднестатистической команды. Возникает вопрос, сколько же команд действительно выдающиеся, не похожие на обычные, стандартные команды, с которыми мы работаем. Время от времени такие встречаются, но, по-моему, не чаще, чем в 10% случаев.

Скотт: **Пусть будет 10%. Как ты думаешь, обладают ли эти команды некими уникальными неповторимыми качествами или есть способы «скопировать» их успех? Допустим, у меня есть компания из сотни сотрудников, которые поделены на 10 команд. Я понимаю, что такое высокопродуктивная команда, и считаю, что в моей компании такая одна. Вопрос: как сделать другие команды похожими на эту, высокопродуктивную? Или это невозможно, поскольку то, что делает команду хорошей, тесно связано с уникальными взаимоотношениями, сложившимися между конкретными людьми?**

Стив: Это спорный вопрос, но, по-моему, нет ничего загадочного в создании высокопродуктивной команды. Понимаю, что это может прозвучать слишком уверенно, но мне кажется, у меня есть для этого основания.

Последние два года подряд мою компанию признавали лучшим работодателем в Вашингтоне среди некрупных компаний. Мы добиваемся этого, потому что у нас очень хороший морально-психологический климат, который мы создали при помощи способов, о которых я собираюсь рассказать.

Если среди вас есть эгоисты, которые думают только о своих желаниях, будет очень сложно создать высокопродуктивную команду. У вас бу-

дет набор отдельных людей, не более того. То есть необходимы «добрая воля» и гибкость. Пусть даже в скромных объемах.

Если у вас есть и гибкость, и добрая воля, то есть рецепт, по которому можно действовать. Во-первых, руководитель должен определить общую концепцию, видение или цель, и для этого, по сути, нужны лидерские качества, а не только умение руководить. Это выходит за рамки функций управления. Лидеры в группе должны очень серьезно продумать эту цель для команды, каким образом следует ее задать, чтобы она по-настоящему, без дураков, вдохновляла и двигала людей вперед.

Классический пример: если вы копаете канаву, ничего возвышенного и вдохновляющего в этом нет. Но если вы роете траншеи, чтобы защитить свой город от надвигающегося врага, воодушевление будет значительно выше, хотя по сути это та же самая работа. И в задачи лидера входит определить или подать работу так, чтобы люди поняли, в чем ее ценность.

Итак, для начала у нас есть добрая воля и вдохновляющая, возвышающая цель или концепция. Следующий шаг – огромное количество общения и взаимодействия по всем направлениям: между членами команды и руководителем, между руководителем и командой. И очень хорошо, если получается упростить всякое взаимодействие внутри группы.

Следующим шагом полезно будет создать чувство командной идентичности, осознания себя как команды, что, в частности, очень хорошо налажено в Microsoft на протяжении уже многих лет. Участники команды получают футболки с символикой, ездят куда-то вместе, получают награды как команды, и так далее, и все в целях формирования этой командной идентичности. Неважно, что это будет – кружки с символикой, майки, куртки, полотенца. Этим вы убиваете двух зайцев. Во-первых, участники команды чувствуют, что их ценят, что очень важно для морального духа коллектива в целом. Во-вторых, вы формируете чувство командной идентичности, что тоже очень хорошо.

Кроме того, есть групповые виды деятельности и мероприятия. Я уже говорил о том, что нужно способствовать неформальным видам общения в команде, но обязательно нужны и разные групповые мероприятия, которые способствуют сплочению команды. Многие смеются над такими вещами и не относятся к ним серьезно, однако всем нравится участвовать. И, по-моему, всем понятно, чего вы хотите этим достичь, однако на деле многие компании отказываются выделить совсем небольшие деньги на подобные мероприятия.

Попробую по-другому: работа и старания по формированию команды полезны сами по себе. Ну, такие вот советы приходят в голову в первую очередь.

В книге «Peopleware»[1] (Dorset House) Том Демарко и Тим Листер недвусмысленно говорят о том, что не знают, как создать команду, но точно знают, как можно развалить команду. Они называют это teamicide – «убийство команды» (от слова team – команда и suicide – самоубийство). Я на практике наблюдал все, что они описывают, – как можно помешать команде сложиться, а если она уже сложилась, как побыстрее ее развалить; и компании делают это постоянно. Берут слаженную и успешную команду, разбивают ее и теряют все преимущества.

Скотт: **Почему это так часто случается? Думаю, любой согласится, что команды необходимы. Что лучшие результаты достигаются за счет структур, объединяющих людей. Работая в Microsoft, я никак не мог понять: почему вся система премий и бонусов построена на индивидуальной основе? Почему структура оценки работы, от чего зависит получение премии или повышение, основана только на твоих личных показателях?**

При этом они, действительно, пытаются создать командный дух. Организуют общие мероприятия, есть одежда с символикой – все то, что может сплотить людей. Но я ни разу не работал в команде и не видел такой команды, где была бы установка, что если мы как команда выполняем сроки, каждый получает премию 2%. Моя извечная жалоба и досада заключалась в том, что система поощрения там никогда не соответствовала тому, что заявлялось на словах.

Они все время говорят о важности команд, но когда приходит время оценивать показатели работы, грант на получение акций и прочие блага достаются лучшему программисту. Почему между теорией и практикой команд всегда такая пропасть?

Стив: Ну, бывает, что они не так сильно расходятся. Мы работали с некоторыми командами, которые поощряли сотрудников на командном уровне, а не делили команды на отдельных людей. Но насколько это работает – остается вопросом.

Но я понимаю, о чем ты говоришь, и видел такое. Джеральд Уэйнберг называл это «плохой конгруэнтностью».

Скотт: **Да.**

Стив: На словах у вас одно, на деле другое. В целом, я согласен с тем, что чем более последовательным и согласованным будет ваше поведение, тем лучше.

---

[1] Демарко Т., Листер Т. «Человеческий фактор: успешные проекты и команды». – Пер. с англ. – СПб.: Символ-Плюс, 2005–2009.

Скотт: **Вот еще одна мысль. Если высокопродуктивных команд так мало, может быть, есть другие способы создавать их? Такие, которыми обычно не пользуются. Когда уже есть очень успешная команда, что часто бывает у разработчиков ПО, и проект завершен, команду чаще всего распускают. И эти ресурсы распределяют по другим проектам. Но можно принять альтернативную модель: успешная команда — это ваш самый ценный актив, и его нужно сохранить, найти способ перекинуть эту команду целиком на новый проект. В таком случае пренебрегаешь знанием предметной области работы, оптимальным распределением ресурсов, только чтобы сохранить команду. Ты встречал когда-нибудь ситуации, когда команда стояла в приоритетах руководства на первом, а не на последнем месте, как это обычно бывает? И вне зависимости от того, видел ли ты такую модель, она не кажется тебе смешной?**

Стив: Забавно, что я сейчас буду противоречить сказанному мною же только что, но многие компании очень ценят знание предмета. Одним их побочных эффектов этого становится их нежелание переводить человека из одной области работы в другую, если он уже наработал в ней опыт.

И хотя в командах со временем происходят перестановки, если человек начал работать в одной области, то эта проблема возникает гораздо чаще. То есть проблема в том, что люди начинают скучать в своей области и мечтать о новых возможностях, а компания не желает их переводить, потому что сделала вложение в данную конкретную область.

Это встречается гораздо чаще, чем ситуации, когда компания перекидывает сотрудников в непредсказуемом направлении, когда проект завершен, или вовсе теряет ценный опыт этих людей. Но тут еще надо сказать, что не так часто сталкиваешься с такими сплоченными и успешными командами, чтобы делать выводы о том, как компании обычно поступают с ними; мне нужна гораздо более репрезентативная выборка.

Скотт: **Хорошо, вопрос был не вполне корректен. Но, как мне кажется, многие из тех, кто работал в хороших командах, согласятся с тем, что их не очень берегли. Система поощрения и укрепления таких команд обычно недостаточна, и они очень часто разваливаются. Я просто подумал — а что, если пойти от противного: сделать ставку на то, что крепкая и успешная команда является самым важным активом, и затем уже выстраивать работу вокруг этого принципа?**

Стив: По-моему, Демарко и Листер писали об этом. Кем-то была предложена идея о том, чтобы команды могли иметь возможность участвовать в тендере на выполнение работы внутри компании. Идея была следую-

щей: команда заявляла, что если ее оставят в текущем составе, то они берутся выполнить проект за четыре месяца. Другая команда могла заявить, что они готовы сделать эту работу за пять месяцев. То есть делается условная оценка возможностей команды в ее текущем составе и прикидывается, выигрываете ли вы от того, что они остаются вместе, и если команда заявляет, что для них столь важно работать вместе, что они готовы работать больше, чтобы выйти на хороший уровень в этой области, то для компании это может стать весьма привлекательной моделью.

Мне кажется, слабое место здесь в том, что команда может заявлять о чем угодно, но это вовсе не значит, что обещания будут выполнены.

Скотт: **В таком случае еще и создаются условия для того, чтобы команды соревновались друг с другом за проекты.**

Стив: Это возможно. Конечно, я могу анализировать такую ситуацию с чисто теоретической точки зрения, потому что ни разу не сталкивался с подобной практикой на деле. По-моему, такая модель потенциально может привести к тому, что людям придется работать в очень жестких условиях. Можно представить себе следующий сценарий: команда работает вместе, проект получается замечательно, они берут на себя очень жесткие обязательства по своему второму проекту, а после его завершения добровольно распускают команду, потому что не хотят больше попадать в такие жернова.

Скотт: **А может ли высокопродуктивная или просто хорошая команда создать плохую программу и наоборот, слабая команда — разработать замечательную программу?**

Стив: Тут многое зависит от того, что вы понимаете под «хорошей» и «плохой» командой. По-моему, есть немало команд, где все сотрудники имеют хорошую мотивацию и им нравится работать вместе, но они совершенно оторваны от деловых реалий; то есть самой команде кажется, что они очень хороши, однако заказчик и рынок так не думают.

Скотт: **Чем-то похоже на твой студенческий проект, да? Команда, у которой есть свои цели и они их замечательно достигают, но с точки зрения руководства или компании они не очень успешны.**

Стив: На одном из моих первых консалтинговых проектов президент компании, который не имел технического опыта, вызвал нас к себе и сказал: «Я купил эту компанию, занялся активным управлением, и у меня назначена дата релиза. Мои технические специалисты постоянно твердят, что успеют к ней, но я уже не верю им. Если честно, по-моему, ребята пребывают в эйфории от своих технологий и им наплевать, выпустим ли мы продукт вообще».

Первой моей реакцией на его слова была: ну вот, еще один классический представитель делового мира, совершенно не чуткий к своим сотрудникам. Уверен, что его ребята вовсю пашут, а он, конечно, не понимает, насколько много им приходится работать. Мы углубились в вопрос, провели много бесед с сотрудниками, и оказалось, что этот человек был прав! Команда разрабатывала ПО для спецэффектов в киноиндустрии. Технологии были невероятными, и ребятам было действительно наплевать, смогут ли они продать эту программу или нет. У них не было никакого чувства деловой необходимости или желания выпустить прибыльный продукт, они замечательно проводили время в работе над ним. По-моему, вот великолепный пример случая, когда команде казалось, что она очень хороша, тогда как президент думал о них иначе, и эта хорошая команда могла в конечном итоге привести компанию к банкротству.

Скотт: **А в обратном случае, может ли слабая или недружная команда, команда, которой все недовольны, да они и сами недовольны друг другом, – могут ли они разработать хорошую программу?**

Стив: Думаю, да. Но только потому, что, как говорится, раз в год стреляет и незаряженное ружье. Командный дух и мотивация обычно идут рука об руку, хотя и не всегда. И различие между этими двумя понятиями я усвоил уже очень давно. Во многих случаях их можно объединить, однако бывает и так, что при высокой мотивации командный дух и дисциплина оставляют желать лучшего. Или, наоборот, при хорошей атмосфере в команде мотивация очень низка или ошибочна.

Примеры этому можно найти в спорте. Самый распространенный пример – это то, что есть очень успешные команды с прекрасными отношениями между игроками, но время от времени обязательно встречаются столь же успешные команды, где люди ненавидят друг друга. Не то чтобы это частый случай, но такое встречается.

Есть успешные команды, состоящие из сильных, выдающихся личностей. А бывает, что команды с выдающимися игроками с треском проваливаются. Бывает и так, что в очень успешной команде нет настоящих звезд, зато ее участники могут по-настоящему хорошо и слаженно работать вместе.

А есть и обычные средние команды. Нет ни ярких личностей, ни выдающейся командной игры. Все это относится и к командам разработчиков.

Скотт: **Вы изучали информатику и вычислительную технику?**

Стив: Да.

Скотт: **Какая часть учебной программы — проекты, над которыми вы работали, полученные задания — имела отношение теории или практике командной работы?**

Стив: У меня диплом бакалавра философии, и...

Скотт: **В философии командная работа не часто встречается.**

Стив: *(Смеется.)*

Скотт: **Увы.**

Стив: Может быть, там ее даже больше, чем в информационных технологиях.

Скотт: **Надо же.**

Стив: Информатика была у меня только второй дисциплиной, и я до сих пор не понимаю, как мне вообще начали поручать эти большие проекты с многочисленным персоналом. Когда я пошел в магистратуру, одной из моих целей было получить больше опыта и научиться лучше работать в команде. И одним из преимуществ программы магистратуры Университета Сиэттла в области информационных технологий для меня было то, что она предполагала много командных заданий. Не только итоговый годовой проект готовился совместно, но и большинство заданий, особенно крупные и сложные, в большинстве дисциплин выполнялись в группах.

Скотт: **Тебе это помогло понять, как устроена работа в команде?**

Стив: Конечно, и, раз уж ты упомянул об этом, мне кажется, было бы полезным, если бы в программе предусматривалось разъяснение того, как успешно выполнять командные задачи, в отличие от индивидуальных, но, в любом случае, у нас было много хороших находок и открытий. Хотя, по-моему, другие группы студентов могли так к ним и не прийти.

Скотт: **У меня в Университете Карнеги Меллон было точно так же. На предпоследнем курсе у нас начались проекты, которые предполагалось выполнять в командах, но в большинстве случаев команды формировались на базе уже существовавших дружеских отношений. Никто не направлял нас даже в таких простых вещах, как определение пяти самых частых конфликтных ситуаций, которые могут встретиться в команде. И нужно-то было всего полчаса, краткое обсуждение в классе пяти самых распространенных видов конфликтов и способов их разрешения. Но ничего такого не проводилось. И этот вопрос я задал потому, что есть большие различия между тем, чему нас учат, и тем, с чем мы сталкиваемся потом. Учат тому, что ты программист-**

одиночка и цель твоя – самому стать мастером в деле программирования, однако приходя работать, попадаешь в условия, где все построено на командной работе.

Стив: Мне кажется, всегда есть эта разница между твоим ремеслом, тем, что ты в нем ценишь, и тем, чему учат в академические часы. Соглашусь с тем, что приведенный тобой пример был бы очень полезен и в моей ситуации. Пять вещей, которые необходимо знать, чтобы проект шел гладко, – очень бы пригодились. Пять способов запороть проект, пять советов для хорошей командной работы – очень полезные знания. Не думаю, что это применимо только к разработке ПО. Несколько лет назад я участвовал в одном судебном разбирательстве, и адвокат противоположной стороны принял несколько решений, которые, на мой взгляд, были чудовищными ошибками. И, хотя эти ошибки были в мою пользу, не думаю, что его клиентам они помогли. Я поговорил тогда со своим приятелем, который закончил Гарвард в области юриспруденции, и спросил у него, что он про это думает. Часто ли бывает, что мой адвокат принимает что-то во внимание, а адвокат противника упускает это из виду. Ответ приятеля удивил меня.

Он сказал: «Один из необходимых шагов, когда собираешься предстать перед судьей или комиссией судей, с которыми прежде не работал, – пообщаться с людьми и выяснить, что за человек этот судья. Узнать о нем все, что можно. Узнать, на что лучше делать упор, и выяснить, какие решения он выносил в прошлом. Просмотреть все его решения по аналогичным вопросам. Чтобы когда ты предстанешь перед ним, ты знал, чего от него ждать». Я спросил: «Вас этому в Гарварде учили?» В ответ он засмеялся и сказал, что таким практическим вещам там не учат.

А ведь в итоге они могут сыграть решающую роль в том, выиграешь ты дело или проиграешь. Мне кажется, любая академическая программа должна выстроить баланс между теорией и практикой, есть черта, за которой программа становится слишком практической. Тогда это уже не академическое образование, а программа техникума или ремесленного училища.

С другой стороны, хорошо бы, если бы существовали студии, лаборатории или другие подобные места, где можно было бы приобрести практический опыт, и, возможно, мы говорим сейчас скорее о том, что в мире ПО есть потребность в такого рода обучающих студиях, слабо удовлетворяемая в настоящее время.

Скотт: **По-моему, у вузов есть определенные обязательства. Если ваши студенты попадают в условия, где они сталкиваются с чем-то впервые, и вы знаете, что всю свою дальнейшую профессиональную жизнь они будут сталкиваться с такими вещами, вы обязаны как ми-**

нимум предложить им: «Если у вас проблемы с работой в команде, почитайте такой-то сайт, такую-то книгу или приходите на такой-то семинар». В моем опыте обучения – хотя учился я уже 15 лет назад – ничего такого не было. Эти вопросы даже не поднимались. Можно было пойти к своему научному руководителю, но он в принципе не хотел слушать о каких-то твоих социальных проблемах. Ко всему, что не касалось самого написания кода, отношение было в духе грубого армейского «справляйся сам». Это подчеркивало ту идею, что твоя успешность как программиста зависела от твоего умения работать с компилятором, а не с людьми или в команде.

Допускаю, что это разные умения. Умение хорошо программировать не связано напрямую с умением работать в команде, можно быть сильным в первом и полным нулем во втором. Но когда речь заходит о прекрасных командах, я в первую очередь спрашиваю у людей о том, какой была их первая команда, как они учились работать вместе, как пришли к понимаю того, что такое хорошая команда. У них должно быть хотя бы базовое понимание того, как все будет работать, и общие понятия о том, как нужно себя вести, чтобы команда преуспела. Большинство выпускников университетов по специальности ИТ понятия не имеют о таких вещах, и это смешно. Это несерьезно, учитывая фактические условия деятельности профессионального программиста в наши дни.

Ну, я выпустил пар. И все это подводит меня к более практичному вопросу. Как ты думаешь, поощряют ли одни методы разработки работу в команде в большей степени, чем другие?

Стив: Определенно. Есть методы, которые очень помогают формированию команды.

В методе Scrum[1], например, очень полезно ежедневное общение на совещаниях. Есть огромное количество команд, где люди не сталкиваются друг с другом лично каждый день. Кроме того, экспресс-планирование, которое осуществляется обычно на уровне целой команды, примерно раз в месяц, дает команде возможность пообщаться на более глубоком уровне, более целенаправленно. Такие приемы можно применять в любых проектах, но в Scrum их применяют чаще, потому что сам этот подход на них построен.

Мне кажется, в программировании недооценивается польза личного общения. В экстремальном программировании (eXtreme Programming) акцент делался на парное программирование. Сейчас очень немногие

[1] Scrum, eXtreme Programming – гибкие методологии разработки ПО. – *Прим. перев.*

компании придерживаются этого подхода. Компании очень выборочно применяют этот метод, и что больше всего нужно программистам, это больше общения и больше взаимодействия.

Очень распространенное сейчас явление – географически удаленные команды (multisite teams). Конечно, многое можно сделать, чтобы добиться лучшего от команд, разнесенных по часовым поясам, но это будет именно попытка слепить лучшее из того, что есть. Практически неизвестны случаи, когда такая структура проектов дает чистое преимущество. Если они работают очень-очень хорошо, максимум, что вы получаете, это отсутствие чистых убытков.

Скотт: **Встречали ли вы высокопродуктивные виртуальные команды? Под виртуальными я понимаю команды, в которых большинство участников не встречаются лично.**

Стив: Не вспомню ни одной. Тут еще другой вопрос: что, если команда разбросана по двум или трем разным офисам или разным часовым поясам? Можно неплохо справляться с двумя офисами. Но если их больше двух, достичь эффективности в работе гораздо труднее. Вы рискуете получить очень низкую результативность, огромную кучу проблем, связанных с плохим взаимодействием, и массу работы по исправлению таких коммуникационных ошибок.

Скотт: **Я уже спрашивал о лучшей команде в твоей карьере. А какая была худшей? Хочется услышать страшилку, чтобы как-то уравновесить истории об успешных командах.**

Стив: Думаю, мне сильно везло. Не могу припомнить ни одной команды, которую бы я считал плохой. Многие могу охарактеризовать как средние, невыдающиеся, но не было случаев, чтобы я считал всю команду плохой. Бывало, пара человек в команде значительно портили лично мне удовольствие от работы в ней, но говорит ли это о том, что команда плоха?

Интересный вопрос: может ли один человек развалить команду или привести ее к провалу? Мой опыт говорит – да. Все зависит от того, насколько это отрицательный персонаж: если это человек действительно вредный и неприятный, дела могут пойти очень плохо. Я как-то работал в группе, где одна сотрудница очень ревностно относилась к формальностям.

И если вы брали из ее отдела ноутбук с фрагментом кода, который был нужен вам для работы, даже если было запланировано, что вы будете работать над ним две недели, она могла прийти во время обеда, забрать с вашего стола ноутбук и унести его обратно в свой отдел. *(Смеется)* И нужно было снова идти к ней и выпрашивать его.

Он был ей не нужен. Ей нужно было быть уверенной в том, что он не останется у тебя дольше, чем нужно. А когда работаешь над чем-то две недели и за это время вынужден 20 раз сходить за ноутбуком, который достаточно было взять один раз, это очень раздражает. И, конечно, это только верхушка айсберга.

Скотт: **Есть и более сложные ситуации. Когда команда хорошо работает, но есть один человек, который, очевидно, создает больше всех проблем и неприятностей, с ним сложнее всего работать, он не переносит критики, но при этом он, к сожалению, самый талантливый специалист в команде. Что делать в таком случае?**

Стив: Ну, Джеральд Дорсет в своей книге «Psychology of Computer Programming» (Психология компьютерного программирования) (Dorset House) прямо говорит, что если у вас есть незаменимый программист, его нужно уволить.

Впервые я прочитал об этом в 1989 году, и тогда мне показалось, что это слишком радикально. Но с годами я понял мудрость этого совета. Иногда увольнение – слишком жесткая мера. Попробуйте убрать человека из команды или хотя бы отстранить от выполнения критически важных заданий. Причина, как я понимаю, в том, что чем дольше вы будете позволять этому человеку работать в таком режиме, тем выше риск в случае его ухода.

Если провал вашего проекта может зависеть от одного человека, значит вы плохо управляете рисками, и самым низким этот риск является *на сегодняшний день*. Дальше он будет только возрастать, если тянуть, ниже он не станет. И чем скорее вы предпримете действия по его снижению, тем лучше для вас.

Кроме того, я обычно обнаруживал, что человек, который кажется незаменимым, на деле таким не является. Обычно есть люди, которые тихо делают свое дело и приносят куда больше пользы, чем этот сложный в общении, привлекающий к себе внимание человек. Я уверен, что бывают исключения, но в первом приближении этот человек наверняка не столь хорош, как вам кажется. То есть, во-первых, он представляет огромный риск для вашего проекта. Во-вторых, он, возможно, даже не столь уж хорош в деле.

В-третьих, он портит жизнь всем на проекте, и как руководитель я в первую очередь думаю: найди способ его усмирить и приручить, а если не получится, найди способ его выставить. А как владелец компании я думаю: избавься от него как можно скорее, ты зря тратишь на него деньги.

В анализе успешности команд, который проводили Ларсон и Лафасто, есть любопытный момент: они описывают проведенное исследование сотрудников команд и руководителей команд. И руководители отмеча-

ют как свою сильную сторону умение взаимодействовать с проблемными сотрудниками, тогда как участники команд говорят о том, что самая слабая черта их руководителя – чрезмерная медлительность в принятии мер в отношении проблемных сотрудников.

По-моему, это говорит о том, что есть большая разница между тем, как руководитель оценивает эффективность своей работы над такими вопросами, и тем, какова она на самом деле. По-моему, когда руководитель в конечном итоге принимает меры в отношении проблемного сотрудника, реакция команды бывает обычно не «Какая несправедливость!», а «Почему вы этого раньше не сделали?»

Скотт: **И как руководители могут избежать этого разрыва в восприятии своих действий и их реальной ценности?**

Стив: Я работаю в Construx последние двенадцать с половиной лет, и у нас очень четко заявлены главные ценности компании, две из которых – открытость и ответственность за свои действия, и два эти принципа работают в нашей компании гораздо эффективнее, чем я мог видеть в других компаниях. И после того как я двенадцать с половиной лет придерживаюсь этих принципов, я просто не понимаю, как можно руководить компанией или командой без той прочной опоры, которую они дают.

Я привожу их в паре, потому что мне кажется, они не работают по отдельности. Невозможно достичь хорошего уровня ответственности за действия, если нет открытости и полного понимания того, кто чем занимается и насколько эффективно, укладываются ли они в сроки и каково качество их работы. Если все это скрывается, то очень просто вообще избежать ответственности за свои действия.

Нужно делать все эти процессы открытыми. В противном случае приходится слишком полагаться на удачу, не зная, насколько верную информацию получаете. И точно так же, если у вас есть открытость, но плохо с подотчетностью действий, люди начнут думать, что вы не справляетесь со своей задачей, спите за штурвалом, потому что видя, что одни работают очень хорошо, а другие постоянно срывают сроки, вы ничего с этим не делаете. В этом случае открытость станет для вас скорее проблемой, чем преимуществом.

А в плане возможности управлять рабочим процессом и своевременно решать вопросы персонала и кучу всех этих постоянно возникающих мелких вопросов по работе команды, если тебе постоянно приходится решать, открывать ли какую-то информацию или выдать ограниченные сведения, это значительно снижает твою эффективность, потому что половину своего времени ты тратишь на то, чтобы объяснить, кто о чем должен знать и прочие подобные вещи.

Тогда как если вы просто открываете все карты, все становится гораздо проще; я знаю, что это не единственный способ успешно руководить организацией, но для себя лично не вижу, как можно пойти на все огромные издержки, предполагаемые любыми другими способами. Не представляю, как компания может эффективно работать без должного уровня ответственности за свои действия и открытости. По-моему, большинство организаций только на словах отдают должное этим понятиям, а на деле у них нет ничего подобного.

Скотт: **Твои слова наверняка многих напугают, особенно тех, кто стоит у власти. Именно те, кто наверху, должны больше всего бояться быть открытыми и подотчетными в своих действиях, потому что их авторитету угрожает наибольшее количество других людей.**

По моему опыту, очень немногие из тех, кто стремится к власти, в то же время заинтересованы в том, чтобы быть открытыми и подотчетными в своих действиях перед теми людьми, над которыми они хотят стоять. Не то чтобы такого совсем не бывало. Но это редкость. Я работал с менеджерами и руководством, обладавшими теми качествами, о которых ты говоришь, но встречаются они, действительно, редко. Возможно, даже реже, чем высокопродуктивные команды.

Стив: Да, но я просто не знаю, как еще добиться этого. В плане разработчиков ПО любопытно то, что мы говорим здесь в основном о том, чтобы направлять мысли людей. Мы платим людям за то, что они думают, и невозможно стоять над кем-то с кнутом и диктовать ему, о чем думать.

Нужно, чтобы они сами захотели думать о тех вещах, за которые мы платим. И в данном случае программист во вполне конкретном смысле вкладывает в работу свою душу, чего не делает, например, оператор станка или продавец за прилавком – потому что при выполнении такой работы они вольны думать о чем хотят.

Вы хозяин своих мыслей, но как программист вы фактически сдаете кому-то в аренду участок своего мозга, наверняка, значительный, на коммерческих условиях. И в этом случае, когда вы платите людям за то, что они думают, пытаться скрывать от них что-то или обманывать, по-моему, просто не имеет смысла. Ну, может быть, какое-то время это будет работать, но в долгосрочной перспективе вы наверняка проиграете.

Скотт: **Согласен, но это не объясняет всех обычных методов, тех, что видишь в большинстве компаний, с которыми я имел дело. Я согласен с тобой в принципе, но убежден, что большинство компаний очень далеки от описываемых тобой стандартов. Мне кажется, что дело здесь еще и в размере компании. Чем больше слоев и уровней, тем сложнее сохранять общую открытость и ответственность за свои действия. Появляются некие «карманы».**

Стив: Да.

Скотт: **И менеджер может заявить: мой отдел, моя команда, моя группа открыты. Часто в рамках группы есть такие «карманы», где соблюдаются открытость и подотчетность. Но в рамках целой компании или подразделения их нет. Здесь действует принцип клана. Если клан становится большим, очень трудно поддерживать такие вещи.**

Стив: Чем больше вы разрастаетесь, тем это сложнее.

Скотт: **Верно.**

Стив: У нас есть это преимущество, небольшие размеры компании. Однако я работал в нескольких компаниях, столь же небольших, а то и меньше, где было очень много политики. Поэтому нельзя делать вывод о том, что если компания маленькая, то она автоматически будет открытой.

Скотт: **Хорошо, последний вопрос. Допустим, я рядовой сотрудник, программист в команде, которая работает не очень хорошо. Это плохая команда. Если взять список критериев успешности команд, мы не то чтобы не совпадаем по всем пунктам, но по большинству не проходим; мы работаем вполсилы. Мы и функционируем со сбоями, и как команда не очень приятны. Прежде чем принять решение уйти – что я могу попробовать сделать, чтобы исправить ситуацию в команде? Могу ли я что-то сделать?**

Стив: Думаю, да. Вести людей можно как сверху, так и снизу, и большинство вещей, которые я назвал необходимыми для формирования команды, может сделать рядовой сотрудник.

Вы можете создать побуждающую и вдохновляющую концепцию. Понять, почему то, что вы делаете, – важно, и донести это понимание до остальной команды. Можете организовать командные мероприятия, как на работе, так и вне ее. Первая моя команда, о которой я рассказывал, была ярким примером этого, мы сложились как команда скорее вопреки руководству, чем благодаря ему, и на самом деле нас как команду объединяла, пожалуй, общая мысль о том, что наш босс жадный тупица.

То есть, думаю, тут необходимо, чтобы кто-то проявил лидерские способности; я имею в виду, что человек должен понять, куда он хочет идти, и попытаться вести людей к этой цели. Куда вы хотите идти – это одно, а как вы хотите достичь этого – другое.

В технических организациях часто можно увидеть разницу между официальной иерархией подчиненности и неофициальной иерархией экспертных знаний. И во многих случаях эта неофициальная иерархия

влияет на ежедневную деятельность гораздо больше, чем официальная структура подчиненности. Это забавно, потому что есть человек, который не указан ни в каких официальных оргструктурах, но к нему все обращаются с вопросами или полагаются на его мнение в ключевых технических и стратегических вопросах. В большинстве организационных структур в конечном итоге этого человека признают, и он получает место где-нибудь в уголке структуры. Но такое лидерство не привязано к бюджету. Человек может так и не получить никакой должности, но проявлять лидерские качества, и мне кажется, такой человек вполне может изменить жизнь команды к лучшему или к худшему.

Скотт: **Мы общаемся уже больше часа. Какую ключевую вещь о работе команд большинство людей, по-твоему, никогда даже не слышали?**

Стив: Мы слишком редко видим, чтобы кто-то прикладывал усилия или хотя бы думал о том, как создать команду. Методы, о которых я сейчас рассказал, не гарантируют, что вы получите идеальную команду. Но если их не применять, вы почти гарантированно таковой не получите. Не думаю, что компании должны тратить на это большие средства, не думаю, что необходимы специальные знания для того, чтобы команда смогла сработаться. Думаю, даже маленькие шаги могут дать большие результаты.

Я бы сказал, что меньше трети компаний, с которыми мы работаем, имеют какой бы то ни было выделенный бюджет на поддержание командного духа или бюджет на командные мероприятия, и это возмутительно. Те, у кого есть такой бюджет, отводят на эти цели совсем небольшие деньги. Может быть, несколько сотен или пару тысяч долларов в год, и это в сравнении с расходами на зарплату в миллион долларов (и это у небольшой команды будет миллион), а на тимбилдинг надо выделить всего пару тысяч. Это очень маленький процент, но отдача от него очень высока.

17

Воспоминания о проекте повышения продуктивности разработки ПО

Идеальная команда, взявшаяся изменить культуру[1]

Барри Бём и Мария Э. Пенедо

История проекта повышения продуктивности разработки ПО

В начале 1980-х я (Мария) после получения докторской степени в области информационных технологий в Калифорнийском университете (Лос-Анджелес) искала работу. Мне очень нравилось избранное поле деятельности и немного тревожила будущая конкретная работа. Очень хотелось найти такое место, где бы можно было заниматься любимым делом, принося реальную пользу. Познакомившись с Барри на конференции, где выступала с докладом о результатах своих исследований, я пошла к нему в TRW на собеседование. Там Барри описал мне условия

[1] Примечание составителей: те, кому довелось работать в команде разработчиков в последние 20 лет, несомненно, испытали влияние Барри Бёма. Он одним из первых применил систематический подход к оценке и планированию проектов разработки ПО. И многие (как и мы) считают, что современная идея итеративной разработки прямо следует из его революционной спиральной модели (Spiral Model).

работы в компании и рассказал о своем новом проекте, целью которого было радикальное изменение способов разработки ПО в компании.

Барри сказал, что провел исследование продуктивности разработки ПО [1] в компании, по сути экономический анализ с целью определить, оправдаются ли крупные инвестиции в средства повышения продуктивности разработки ПО. Собственно проведение такого анализа было вызвано тем, что компания собралась выйти на достойный уровень в отрасли и на международном рынке, а также ростом спроса на программное обеспечение, нехваткой специалистов в этой области, растущей потребностью в технических средствах разработки ПО и сокращением расходов на аппаратное обеспечение – звучит очень знакомо и сегодня, не правда ли? Это исследование привело его к проекту, к которому он хотел привлечь и меня, – Проекту повышения продуктивности разработки ПО (ППР) [2, 3]. Он объяснил, что будет непросто, потому что мы будем пытаться *изменить культуру*. Для моей первой работы в данной сфере это звучало очень заманчиво, я приняла предложение и начала работать в проекте вместе с фантастической командой новичков и ветеранов TRW. Наша задача заключалась в попытке изменить текущие методы разработки ПО путем ввода среды поддержки в той части компании, что занималась заказами для авиакосмической промышленности.

Прежде чем влиться в корпорацию Northrop Grumman, TRW была конгломератом из крупной компании – производителя запчастей для автомобилей (Thompson Products в Кливленде) и крупной авиакосмической компании (Ramo-Wooldridge в Лос-Анджелесе). Она также выпускала несколько линеек преимущественно программных продуктов, таких как TRW Credit Data, и включала подразделения, производившие системы для торговых терминалов, коммутаторы и системы управления технологическим процессом. В середине 1980-х TRW была компанией номер два в мире по продажам ПО в ежегодных списках Datamation 100, с большим отставанием от IBM.

В 1960-х авиакосмический сектор TRW стал пионером в переходе разработчиков ПО от подхода «кодировать-и-править» к каскадной модели на основе требований, описанной в классической работе Винстона Ройса [4]. В 1970-х TRW разработала целый ряд регламентов, стандартов, процедур проверок, обучающих курсов и программных инструментов, ориентированных на каскадную разработку. Они очень четко соответствовали техническому, научному уровню и системам управления в TRW на тот момент, и вся культура разработки стала ориентированной на каскадную модель. К концу 1970-х, однако, сфера применения продуктов, разрабатываемых TRW, стала требовать гораздо большего взаимодействия с людьми, и каскадная модель уже плохо справлялась с требованиями, возникавшими по мере развития проекта.

В то время поддержка проектов разработки включала менеджеров, координировавших людей и деятельность, секретарей, которые печатали документацию по проекту, организовывали совещания по планированию и выполнению задач по проекту, и разработчиков и инженеров, которые создавали проектную документацию и сами системы. Инструментами разработки пользовались только разработчики, и им приходилось работать в режиме пакетной обработки либо в специальных залах с ограниченным числом терминалов для интерактивной разработки, на основе мэйнфреймов. В проекте ППР мы создавали новые условия разработки, когда у всех участников проекта (руководителей, разработчиков систем, разработчиков ПО, секретарей и специалистов по контролю) были собственные кабинеты с отдельными рабочими станциями, и общались они по локальной сети (ЛВС). Ее архитектура была основана на ОС UNIX и включала как серийное ПО, так и самостоятельно разработанные инструменты для поддержки всех этапов реализации проекта (определение требований, отслеживание, проектирование и разработка, управление формами и автоматизация делопроизводства). На этом новом проекте нам пришлось столкнуться со значительными техническими рисками и рисками конфликта заинтересованных сторон, а также со срочными требованиями, которые невозможно было предвидеть заранее. Таким образом, у нас появилась великолепная возможность применить раннюю версию разработанной Барри спиральной модели параллельной разработки на основе рисков [5]. Эти изменения вызвали конфликт культур в самом проекте ППР между ветеранами, привыкшими к каскадной разработке, и новичками, ориентированными на интерактивную модель. До сих пор вспоминаются жаркие споры и совещания, на которых старые сотрудники TRW говорили: «Но регламент запрещает создание опытных образцов на столь раннем этапе! Создание опытного образца – это кодирование до момента критического анализа проекта».

Проект оказался очень успешным (продуктивность значительно повысилась, возможно, вдвое или больше, за счет повторного использования), но было непросто регламентировать эти изменения, сделать их официальными и убедить персонал. Мы поняли также, что для повышения продуктивности необходима комплексная программа инициатив в нескольких областях и постоянная упорная работа. Хотя этот проект был реализован уже очень давно, истории, которые мы здесь расскажем, актуальны и поныне несмотря на все развитие инструментов и технологий, как в области планирования, так и в плане реализации мер по повышению эффективности. *Менять культуру очень сложно, но еще сложнее не отставать от очень быстрого развития технологий в крупных организациях.*

Воплощая проект в жизнь

Начало начал: при запросе руководства иметь наготове все варианты

Сцена: Офис Боба Уильямса, конец 1979 года. Боб – вице-предизент/генеральный директор подразделения программного обеспечения и информационных систем (2000 сотрудников), одного из 6 подразделений TRW Defense and Space Systems Group (DSSG) (Группы по оборонным и космическим системам TRW). Барри – его главный инженер и менеджер по передовым технологиям в бизнесе.

Боб: Я только что вернулся со встречи генеральных директоров DSSG по поводу повышения продуктивности. Руководство в Кливленде ставит перед собой серьезную цель сделать подразделение по производству автомобильных запчастей более конкурентоспособным относительно японских производителей и хотело бы, чтобы все в TRW сконцентрировались на повышении производительности. Похоже, компания готова выделять деньги на инициативы, касающиеся повышения продуктивности, если они будут обоснованы с деловой точки зрения. Думаю, нам стоит попробовать. Как думаешь, сможешь составить такой проект для нас?

Барри: Конечно. Это вполне укладывается в рамки тех улучшений, о которых мы говорили, но не смогли добиться их финансирования. Наша конструктивная модель оценки стоимости (Constructive Cost Estimation Model, COCOMO) позволяет создать хорошее экономическое обоснование проекта. Она показывает, как растет или снижается производительность при изменении определенных факторов стоимости, таких как инструменты поддержки, срок выполнения работы, повторное использование компонентов и человеческий фактор. В отношении последнего как раз можно применить твои идеи о создании разных путей развития карьеры для наших сотрудников. Можно попробовать нашу технологию локальной вычислительной сети, чтобы люди могли интерактивно работать и общаться через нее. Плюс можно задействовать поддержку инициатив отдела обороны в области автоматического сбора данных. От нас ждут уже полностью проработанное предложение?

Боб: Вообще, мы будем делать это предложение правительству, вот в чем дело. Но, поскольку это внутренняя инициатива компании, спонсоры хотят получить более четкое представление о предлагаемых вариантах, прежде чем согласиться на большие расходы. То есть у нас есть пара месяцев на то, чтобы составить окончательный вариант. Может, проведешь исследование вместе с Рэем Вулвертоном и парой ребят из

отдела сбора данных и составишь проект предложения? И давай задействуем всех сотрудников – пусть каждый выскажется по поводу повышения производительности.

Барри: Отлично. Приступаю к делу, через пару недель жди от меня отчета о ходе выполнения.

Оценка и отбор вариантов: применение спиральной модели

Постепенное развитие, предполагаемое таким проектом, в отличие от прямого обязательства для всей компании, было нашей первой возможностью применить в полной мере спиральную модель разработки ПО [4], чтобы постепенно с каждым новым витком спирали исследовать варианты, уточнять задачи и получать все более высокий уровень понимания и принятия модели руководством и пользователями. После того завершения первого такого цикла был составлен окончательный вариант проекта и получено разрешение продолжить нашу работу, на втором витке предполагались: посещение центров передовых технологий, таких как Xerox PARC и IBM Santa Teresa, проведение анализа архитектуры и рынка ЛВС и рабочих станций, более подробное исследование потребностей руководителей и разработчиков, создание демоверсий и опытных образцов. Были подготовлены три концептуальных варианта внедрения, в зависимости от уровня затрат – 2000, 10 000 и 50 000 долларов на человека, хотя рабочие станции, предлагаемые Xerox PARC, были очень заманчивы.

Сцена: Офис Боба Уильямса, лето 1980 года.

Боб: Хорошие новости! Нас выбрали для разработки предлагаемой нами системы повышения продуктивности разработки ПО. Парни, вы отлично поработали на первом этапе подготовки официальных документов по проекту, что и позволило нам получить такую поддержку. Рассмотрев данные по производительности проектов IBM и AT&T, они согласились на наш вариант в 10 000 на человека с разворачиванием локальной сети, применением недорогого варианта отдельных кабинетов, как в IBM, и системы на основе UNIX, причем не только для программистов, а для всего проекта, и дополнительными подпроектами по созданию методов управления и новых карьерных возможностей. Нам выделяют на проект миллион долларов в год, нужно только представить на проверку прототипы, технические условия и планы и найти проект, на котором мы будет применять все эти нововведения для разработки того, что им требуется. Думаю, я смогу найти для вас хороший экспериментальный проект. Что вам еще нужно, чтобы начать?

Барри: Фантастика! Мне нужна будет помощь в работе с обслуживающим персоналом, чтобы переформировать и проложить сеть по новым отдельным кабинетам, и кто-то должен сообщить специалистам по ЛВС из TRW, что мы собираемся проводить анализ качества и готовности к применению их продукта. Кроме того, нужно будет нанять опытных специалистов по UNIX. В университетах Лос-Анджелеса и Санта-Барбары, думаю, найдется пара человек, которых это заинтересует.

Боб: Хорошо. К следующему собранию подготовь презентацию по всем этим пунктам, а я выделю тебе персонал для этой работы.

Начало работы: сбалансированная команда, принятый экспериментальный проект и прелести UNIX

Проекту повышения продуктивности разработки ПО повезло с тем, что его экспериментальным подразделением стал отдел, являвшийся частью очень крупного проекта по разработке системы реального времени, с открытым для новых идей руководителем и многочисленными исполнителями, которые после некоторого сопротивления оценили возможность иметь выделенные интерактивные рабочие станции и возможность электронной переписки. И у проекта повышения продуктивности разработки ПО сложилась сбалансированная команда из опытных разработчиков TRW и новых специалистов по UNIX. Ключевые эксперты по UNIX из Университета Санта-Барбары (Арт Пистер) и Университета Лос-Анджелеса (Мария Пенедо) оказались очень хороши: они предлагали новые творческие идеи, увлекали ими участников команды, показывали, как при помощи возможностей UNIX можно улучшить автоматизацию, обеспечить быстрое создание опытных версий и разработку как программных инструментов, так и самого конечного продукта, избежав некоторых типичных для проектов проблем. Благодаря каналам, фильтрам и некоторым другим возможностям UNIX мы могли постоянно реагировать на запросы заказчика; благодаря гибкости среды UNIX мы могли предлагать простые решения и реагировать на запросы очень быстро.

Сцена: Одна из первых демонстраций возможностей UNIX сотрудникам экспериментального проекта, 1981 год.

Скептически настроенный сотрудник старой закалки: Все это очень интересно, но может ли она сделать что-то полезное для проекта, например создать специализированный телефонный справочник?

Арт: Конечно. Применяем команду `grep` к именам в списке проекта и именам в телефонной книге компании, затем командой `pipe` направляем их в новый файл... и все.

Рассказы о проекте

Да, мы создали команду, замечательную команду из 20–25 человек. Были у нас и ветераны TRW, но большинству участников было меньше тридцати, некоторые только из университета, бакалавры, люди со степенью магистра или доктора наук, хороший баланс женщин и мужчин, несколько иностранцев (из Южной Америки, Азии и со Среднего Востока) и море энтузиазма. Мы довольно скоро сплотились, вместе бегали обедать (сейчас такое еще бывает?), некоторые даже стали близкими друзьями и вне работы. Конечно же, у нас были и «ботаники», которые выдавали тонны кода и сторонились всех (про них мы расскажем позже). Надо иметь в виду, что большинство членов команды прекрасно разбирались в новых технологиях, поскольку пришли из университетов, которые были пионерами в применении Интернета (например, Калифорнийский университет в Лос-Анджелесе). Однако целевая аудитория представляла собой скорее обычный персонал компаний, привыкший работать «по старинке».

Задачи

Нашей задачей было убедить обычный персонал проекта в необходимости принять новый, командный стиль разработки и сотрудничества и применять предлагаемые нами новые технологии. Целью ППР было изменить культуру, развернуть локальные вычислительные сети и организовать общение всех участников проекта по электронной почте, а также применить автоматизацию в разработке и прочей деятельности проекта. (Не забывайте, что это были 1980-е... С другой стороны, с тех пор мало что изменилось, мы видим в крупных корпорациях те же проблемы, когда опытный персонал не решается принять новейшие технологии, такие как социальные сети, Википедия и виртуальные миры.) Мы работали в сотрудничестве с другим проектом, который должен был использовать новые средства/процессы, созданные нашей командой, и у нас есть много интересных историй об этом сотрудничестве.

Воспитание руководителя

У нас был замечательный руководитель, очень дружественно настроенный и открытый всему новому – в конце концов, он взял на себя непростую задачу руководства таким «революционным» проектом. Но он работал в этой компании уже больше 20 лет и очень привык к существующей производственной культуре. Мы, молодежь, пришли менять его привычки. Он с большим энтузиазмом отнесся к большинству предлагаемых средств автоматизации производственного процесса, но мы заметили, что он никогда не пользуется электронной почтой (в то время никто из руководителей ею не пользовался). И одна наша сотрудница

решила его провести: она записалась к нему на прием, но войдя в кабинет, объявила, что будет учить его пользоваться электронной почтой. Надо было видеть его лицо! Он привык сам говорить сотрудникам, что им делать, а не наоборот. Короче, он не смог отказаться, и после того как его научили, стал полноправным членом команды, участвовал в обсуждениях и конференциях онлайн, вникал в вопросы, делился новостями. Избегая этого нового для него метода, он упускал важную составляющую жизни проекта, его «кухню». Позже он признался, что «не умел печатать» и очень стеснялся этого. Но эта сотрудница и ее начальник стали хорошими друзьями и коллегами в ходе реализации проекта, благодаря его открытости новому. Вот ключевые ингредиенты, способствовавшие успеху того проекта.

Можно мне отдельный кабинет?

Обычное положение вещей в проекте: у руководителей отдельные кабинеты, у остального персонала – отгороженные рабочие места в общем офисе, у разработчиков – зал, где расположены общие компьютерные терминалы. Одна из внедряемых нами новых идей состояла в том, чтобы каждому предоставить отдельный кабинет независимо от статуса и доступ к компьютерам через выделенный индивидуальный терминал, подключенный по ЛВС. Хорошие идеи. Решение о применении ЛВС распространялось со скоростью пожара, можно было в любое время иметь доступ к компьютерам со своего рабочего места. К сожалению, по экономическим причинам идея «каждому по отдельному кабинету» не работала, ни тогда, ни сейчас. Но тогда мы пытались что-то сделать, хотя и знали, что при нехватке пространства наши усилия будут тщетны. Тогда было предложено сделать отсеки настолько маленькими, чтобы согласно политике компании туда нельзя было посадить больше одного человека. И на время проекта это сработало – у всех наших сотрудников были отдельные кабинеты. Это было одно из тех предлагаемых изменений, которые, к сожалению, организация в целом так и не переняла; нам не удалось доказать заявленные преимущества и преодолеть экономическое давление.

Надо добавить, что остальные сотрудники очень завидовали проекту, потому что у нас были отдельные кабинеты, терминалы на столах и прочее. Приходилось как-то выкручиваться, чтобы не усугублять эту зависть, мы постоянно подчеркивали, что мы «подопытные кролики», проект может провалиться, но если он увенчается успехом, то и у них будет все то же, что есть у нас.

Выбор технологий: важность анализа затрат

Выбор технологий не всегда проходил гладко. Например, встал вопрос: выбрать для нашей ЛВС готовое решение либо самостоятельно разрабо-

тать интерфейсный блок. Был продукт, разработанный в нашей компании, который не был еще полностью готов, но всячески проталкивался высшим руководством, что вполне понятно. Мы провели тщательный анализ затрат и тестирование, но понадобился максимум дипломатии, чтобы убедить руководство в том, что будет меньше риска, если мы приобретем серийные блоки. Знакомо, не правда ли?

Обращение проекта «подопытных кроликов» в «новую веру»

Как мы уже говорили, предполагалось, что в проекте будут использоваться новые разрабатываемые инструменты/среда. Идея с развертыванием ЛВС была принята на ура, но новая операционная система (мы внедряли UNIX в производственную культуру, где использовались машины от DEC под ОС VMS) и некоторые новые инструменты нравились далеко не всем. Чтобы еще усложнить нам жизнь, в экспериментальном проекте использовалась операционная система VMS, которая уже шла с хорошими автоматизированными инструментами. Кроме того, некоторые сотрудники выделенного проекта жаловались, что UNIX на машинах VAX работает гораздо медленнее, чем VMS. Это действительно было так, но они упускали из виду тот факт, что это была среда разработки и общая производительность повышалась, потому что люди были на тот момент самым ценным ресурсом (они все еще считали, что машинное время важнее, чем время сотрудников). В ходе реализации проекта одни сотрудники изменили свои взгляды, а другие так и остались приверженцами VMS.

Любопытно, однако, было наблюдать, как много может значить какая-нибудь мелочь для пользователей. Вот два примера, связанных с внедрением новых систем автоматизации. Первый: у нас была секретарь, которая печатала большую часть документации по проекту, взаимодействуя при этом с разными сотрудниками. Документы часто содержали множество формул, для которых ей нужно было оставлять пробелы, а потом впечатывать их отдельно, хотя она и работала в текстовом процессоре. Это было очень неудобно и медленно как для нее, так и для заказчиков этой работы. Узнав, что у UNIX есть инструмент eqn (уравнение), она сразу поверила в эту систему и обратила в эту веру многих окружающих, включая менеджеров и системных инженеров, которые к тому времени уже научились пользоваться текстовым процессором. Другой пример – технически очень грамотная сотрудница, административный помощник/администратор данных; она выяснила, что в UNIX есть функция scripts, позволяющая автоматически создавать разные бланки и формы, и новые инструменты управления конфигурацией. Ее руководитель был настолько впечатлен результатами, что в приказном порядке обязал весь свой персонал использовать эту систему и инстру-

менты. Она применила творческий подход к решению задачи, а взгляды ее руководителя не помешали ему внедрить это административно (иногда только так можно заставить людей принять изменения).

Стандартизация

У внедрения технологий много разных аспектов, но мы выяснили, что даже очень хорошую технологию иногда можно внедрить только путем стандартизации или в приказном порядке. Так было и в нашем проекте, когда новые средства уже доказали свою эффективность, и это до сих пор случается в разных организациях. Примеры включают стандартизацию использования ПК, продуктов Microsoft; а также SharePoint, eRoom, Livelink, и так далее. Так решило руководство, и пользователям осталось только принять это решение. С другой стороны, даже лучше проводить стандартизацию на уровне интерфейсов, где это возможно, особенно в компаниях вроде TRW, создающих собственное ПО, которые должны поставлять программы и инструменты, совместимые с разными пользовательскими средами.

Пользователи должны быть членами команды

Внедрить новую технологию проще, если пользователи заинтересованы в успехе этой технологии. Если они вовлечены с самого начала и принимают участие в определении требований, анализе, процессе проектирования или если они участвуют в процессе анализа ПО по мере его разработки, то у них возникает чувство собственности в отношении этой технологии, и они скорее будут защищать, чем критиковать или отвергать ее. Этот урок дался нам дорого. Некоторые члены команды собирались не спеша разрабатывать совершенные инструменты и были разочарованы, узнав, что пользователи предпочли бы простые (для UNIX) вещи вроде специальных панелей на полях, указывающих места, где в документ внесены изменения. Но они приободрялись, видя, как их нововведения облегчают жизнь пользователей проекта.

Добиваться общности

В каждом проекте есть свой одинокий волк, который по неизвестным команде причинам не желает влиться в нее, не умея общаться. У нас был такой человек, невероятно умный и талантливый, но совершенно не способный проявлять терпение по отношению к «обычным» людям. Он очень быстро думал и работал и постоянно удивлял нас новыми идеями. Помню, как однажды он ускорил работу пакета программ управления формами в 4–5 раз, полностью погрузившись в UNIX, чтобы добраться до каких-то непонятных вызовов интерфейса. Ему всего-то сказали, что пакет у многих работает слишком медленно. Он проворчал что-то вроде «радуйтесь тому, что есть», ушел к себе и вернулся через

пару дней весь сияющий и с новой версией, которая работала в 4–5 раз быстрее... а мы даже не знали, что он над ней работает.

Но эта его отстраненность была очень тяжела для многих. Нас это тревожило, нам хотелось видеть его в своем кругу, мы чувствовали, что можем действительно улучшить сотрудничество в команде, поскольку он был одним из самых грамотных специалистов среди нас. Не знаю точно, что привело к переменам, но одна участница команды, умнейшая дама с дипломами по музыке и информационным технологиям, сумела стать его другом и пробиться через этот панцирь. Нам кажется, после этого он начал доверять нам и получать удовольствие от работы с некоторыми членами команды. Атмосфера проекта значительно улучшилась. Из этого можно сделать следующий вывод: не оставаясь равнодушными и не пожалев времени, можно найти способ включить в команду любого.

Обучение: руководители – не обычные студенты

Мы запустили на проекте обучающий курс по работе с операционными системами и их многочисленными инструментами. И тут мы обнаружили, что менеджеры не приходят на занятия. Им просто некогда или они не решаются показать, что недостаточно много знают или не очень способны? Мы выяснили, что для многих оказалась верной вторая причина. Чтобы не рисковать, мы организовали отдельные занятия для руководства (они не отличались по содержанию, разве что были чуть меньше детализированы), и среди равных они чувствовали себя более уверенно, а ближе познакомившись с технологией, с большей вероятностью начинали ее использовать, рекомендовать или внедрять в приказном порядке.

Пробовать свои блюда

Очень хорошим уроком для нас стало то, что мы научились в проекте ППР использовать те инструменты и средства, которые разрабатывали сами. Поначалу это не делалось, и мы не понимали некоторых нужд пользователей, что вредило внедрению. Как только проект начал использовать эти средства, например возможность отслеживать требования, мы стали лучше понимать интересы пользователей и таким образом смогли во многом улучшить эти инструменты, что, в свою очередь, вело к тому, что и пользователи легче принимали новую технологию.

Электронная почта: помощь или препятствие?

Что бы мы сегодня делали без электронной почты? (Фактически сегодня роль электронной почты исполняют службы мгновенного обмена сообщениями Instant Messengers и социальные сети.) Мы сделали обязательной практикой обмен электронными сообщениями между

проектами при помощи новой среды. Разумеется, при этом возникали проблемы, актуальные и сегодня. У электронной почты много преимуществ: она позволяет людям общаться и одновременно рассылать сообщения многим адресатам на большие расстояния, позволяет оставлять сообщения, когда получателя нет на рабочем месте, и так далее. Однако она может и отрицательно сказаться на производительности, поскольку вместо того чтобы поднять трубку и позвонить человеку, чтобы решить проблему или обсудить какие-то идеи, человек машинально пишет электронное письмо. Это может занять больше времени для обеих сторон, решение вопросов затянется, тогда как было бы вполне достаточно короткого звонка. Часто раздражало, что человек не отвечает на заданный в письме вопрос, потому что невнимательно просмотрел свою почту или неправильно понял письмо.

Кроме того, проблема отправки зашифрованных писем тоже может вызвать непонимание и неприятные ощущения. Поскольку общение по электронной почте по своей природе не «личное» и не интерактивное, «качество» общения часто страдает. Растет недопонимание, для разрешения которого порой требуются значительное усилия и время. Побочный эффект от внедрения среды проекта ППР состоял в том, что отдельные сотрудники перестали выходить из своих кабинетов, чтобы пообщаться друг с другом. Они посылали зашифрованные сообщения, вместо того чтобы что-то спросить или обсудить, заглянув в соседний кабинет. Это привело к тому, что некоторые вопросы уровня команды мы начали решать на общем собрании, где руководители поощряли команду «разговаривать», а не полагаться всецело на электронную почту в качестве средства коммуникации.

Кроме того, электронная почта не очень эффективна в плане постановки задач и назначения ответственных по важным вопросам. Никогда нет полной уверенности, что ваше письмо правильно поняли, и даже в том, что его прочли. А иногда чрезмерное упование на возможности электронной почты может стать причиной ошибок и неэффективной работы – например когда функцию «ответить всем» применяют без разбора или случайно.

Превращение в текстовые процессоры

Также отрицательно сказаться на производительности могут такие инструменты, как текстовые процессоры (например, LaTeX и другие). Хотя возможность самостоятельно печатать свои документы сильно упростила эту задачу, пользователи были недовольны, что в результате они стали слишком много времени тратить на оформление (правильное форматирование), а не на содержимое документа. Поскольку создание документов легло на плечи самих инженеров (кроме менеджеров, у которых были секретари), стоимость часа такой работы в итоге воз-

растала, а содержимое страдало, потому что больше времени и усилий уходило на то, чтобы правильно отформатировать документ. Конечно, появление текстовых процессоров WYSIWYG[1] решило эту проблему (ну, почти).

Проблемы новаторства, ограниченных сроков и коммерческого применения

На основании потребностей проекта была создана система управления формами, включавшая многие возможности сегодняшних СУБД четвертого или пятого поколения; в то время подобных систем еще не существовало. Это была замечательная система, которой можно было бы найти коммерческое применение, если бы наша компания выходила на коммерческий рынок. Этого не произошло, поскольку мы были (и остаемся) крупной авиакосмической компанией, которая работает в основном с заказами. не забудем, что в команде было много молодых предприимчивых и очень креативных молодых людей, которые расцвели в среде, обеспечиваемой столь инновационным проектом. В результате отсутствия интереса в коммерческом применении кто-то из этих ребят ушел из компании – одни с целью попробовать себя в новых проектах, другие для того, чтобы самостоятельно разрабатывать инструменты, допускающие коммерческое применение.

Железный закон поддержки ПО

Железный закон поддержки ПО гласит: на каждый вложенный в разработку доллар приходится вложить два доллара в поддержку жизнеспособности продукта на весь срок его службы Система повышения продуктивности разработки ПО начиналась как бесплатный сервис для проектов, и заставить проекты платить за дальнейшую поддержку оказалось очень непросто. Поэтому со временем у проектов оставалось все меньше ресурсов, позволявших добавить новую функциональность. Например, система управления формами на момент ее разработки в 1982 году была лучшей в мире системой этого класса, и в 1986 году она все еще оставалась лучшей в мире системой 1982 года в своем классе, однако к тому времени ее уже затмили коммерческие системы управления формами на более широкой финансовой и пользовательской основе.

Сторонники

На этом проекте и на многих других проектах мы понимали необходимость сторонников среди разработчиков и пользователей – тех, кто при-

[1] WYSIWYG (англ. What You See Is What You Get – что видишь, то и получишь) – технология, при которой материал в процессе редактирования выглядит в точности так же, как и конечный результат. – *Прим. перев.*

нял новшество и вдохновился технологией. Все мы знаем, что иногда увлеченность, страсть может сдвинуть горы. И во многих случаях без таких сторонников положительная динамика может сойти на нет. Однако быть сторонником очень непросто, многое зависит от потребности в этой технологии, экономических факторов и руководства. Сторонники всегда находятся в самом центре перемен, и их нужно холить, лелеять и вознаграждать по заслугам.

Проблемы быстрой смены технологии

Главной причиной запуска этого проекта была необходимость внедрения новых автоматизированных средств в компании для повышения производительности разработки программного обеспечения. Однако осознав, что новые технологии внедряются слишком быстро, мы должны были создать возможность смотреть в будущее, чтобы достичь целей проекта, постоянно предлагая и внедряя новые технологии (это было задолго до того, как информационные технологии заняли стали играть столько важную роль для предприятий). И это было (да и сейчас остается) нетривиальной задачей. Параллельно с программой развернулась деятельность по обновлению всей техники, от рабочих станций мы перешли к растровым дисплеям, настольным ПК и так далее. В ходе развития проекта и внедрения других усовершенствований мы обнаружили, что в большинстве случаев самостоятельно разработанные технологии и инструменты быстро устаревали, и их неизменно вытесняли серийно выпускаемые продукты. Каждый шаг начинался с больших надежд, двигал дело вперед и приводил к изменениям и усовершенствованиям, но организациям необходимо постоянно перестраиваться. Многие из предпринятых нами инициатив замечательно послужили тому, что открывали для пользователей новые концепции и демонстрировали необходимость и преимущества технологий, но многие в итоге затухали в результате действия Железного закона поддержки ПО и относительно малой группы пользователей, которым нужны были изменения в самостоятельно разработанном компанией ПО. По этой причине в нашем мире и в наше время большинство используемых технических средств приходят из мира серийного производства, а там компании сами решают сложные задачи устойчивости, стоимости, адаптации к нуждам пользователя и необходимой эволюции.

Изучение изменений в поведении пользователей

Оба руководителя проектов – проекта по повышению производительности разработки ПО и экспериментального пользовательского проекта – высказались о возможности электронной связи сгладить традиционную иерархию подчиненности. Комментарии были примерно такими: «В предыдущем проекте я никогда не знал, что происходит двумя

уровнями ниже, и всегда чувствовал, что слишком поздно реагирую на проблемы проекта. Теперь с большим потоком электронной почты я гораздо более оперативно получаю информацию о возможных проблемах и более четкую картину работы каждого участка проекта». Кроме того, тот факт, что всю созданную документацию и переписку теперь можно было анализировать при помощи электронных средств, означал, что мы лучше могли определять и устранять «узкие места» в таких процессах, как управление изменениями и исправление дефектов. Ключевым преимуществом личных кабинетов в сочетании с увлеченными сотрудниками была возможность достичь такого уровня производственного процесса, о котором говорили Том Демарко и Тим Листер в книге «Peopleware» (Человеческий фактор) [6]. В ходе реализации проекта сам собой появился код дверей. Закрытая дверь означала: мне нужно сосредоточиться, открытая дверь приглашала войти и пообщаться. В результате происходило вот что:

Сцена: Офис проекта повышения производительности, 16.00.

Входит Стив со словами: «Я проголодался. Никто не хочет пообедать?»

Сью отвечает: «Стив, уже четыре. Ты программировал целый день».

Стив: «Да, когда тебе весело, время пролетает незаметно».

Заключение

Проекту повышения продуктивности разработки ПО повезло с некоторыми аспектами, определяющими успешность проекта: поддержка высшего руководства, способная и увлеченная команда, реалистичные графики и бюджет, соответствие требований и разработки решений, итеративное развитие. Однако многие проекты могли похвастаться теми же факторами, но при этом так и не пришли к *идеальной команде*. Если сравнивать проект повышения продуктивности разработки ПО с некоторыми из этих проектов, мы бы назвали следующие факторы *идеальной команды*:

- Определение и вовлечение всех заинтересованных сторон, необходимых для успешной работы
- Большая предварительная работа по сбору информации, исследованиям и формированию команды
- Разработка единой для всех концепции продукта и ее результаты
- Определение руководителя, открытого переменам, хорошо умеющего слушать и обладающего навыками укрепления команды
- Поощрение креативных идей, поступающих извне или возникающих внутри группы

- Чуткое отношение и реагирование на потребности команды
- Дипломатичное устранение несовместимых исполнителей
- Проведение переговоров по конфликтам заинтересованных сторон и достижение компромисса
- Тщательное отслеживание хода работ и раннее предотвращение угрозы провала

Литература

1. Boehm, B., M. H. Penedo, et al. «A Software Development Environment for Improving Productivity», *Computer Magazine*, May 1984, pp. 30–44. Also in R. Selby (Ed.), «Software Engineering: Barry W. Boehm's Contributions to Software Development, Management, and Research», IEEE Computer Society Press/Wiley Interscience, 2007, pp. 245–268.
2. Bitar, I., M. H. Penedo, et al. «Lessons Learned in Building the TRW Software Productivity System», *Proceedings of Spring CompCon*, San Francisco, February 1985.
3. Penedo, M. H. and E. D. Stuckle «TRW's SEE Saga», *Proceedings of the International Workshop on Environments*, Chinon, France, in *Lecture Notes in Computer Science*, No. 467, Springer-Verlag, September 1989.
4. Royce, W. W. «Managing the Development of Large Software Systems: Concepts and Techniques», *Proceedings*, WESCON, August 1970.
5. Boehm, B. «A Spiral Model of Software Development and Enhancement», *Computer Magazine*, May 1988, pp. 61–72. Also in R. Selby (Ed.), «Software Engineering: Barry W. Boehm's Contributions to Software Development, Management, and Research», IEEE Computer Society Press/Wiley Interscience, 2007, pp. 345–365.
6. DeMarco, T. and T. Lister «Peopleware: Productive Projects and Teams», Dorset House, 1987 (2nd Edition, 1999).[1]

Благодарности

Выражаем признательность авторам и редакторам этой главы, включая Кристин Шу Франка Бельца, Арта Пистера и других; всем, кто участвовал в создании и стал частью этой замечательной команды. Мы хотели бы также посвятить эту главу памяти Дона Стакла, менеджера проекта повышения продуктивности разработки ПО и воистину прекрасного человека.

[1] Демарко Т., Листер Т. «Человеческий фактор: успешные проекты и команды». – Пер. с англ. – СПб.: Символ-Плюс, 2005–2009.

18

Создание космических кораблей

Интервью с Питером Глюком

Найдется ли в США гик, который отказался бы работать в НАСА? Мы-то уж точно не отказались бы. Поэтому мы так обрадовались знакомству с Питером Глюком, который работал в Лаборатории реактивного движения уже 20 лет, руководя командами программистов. Нас интересовало, каково работать в команде НАСА и отправлять свои программы в космос, где они должны работать, а ты не можешь просто прийти и устранить какую-то проблему.

Эндрю: **Расскажите немного о своем прошлом: как вы попали в НАСА, над какими проектами работали.**

Питер: Я магистр аэрокосмического приборостроения и бакалавр математики. В четвертом классе я был на экскурсии в Лаборатории реактивного движения здесь, в Пасадене, потом вернулся домой и сказал: «Мама, я хочу работать в ЛРД, когда вырасту».

Эндрю: **То есть вы хотели работать в НАСА с самого детства.**

Питер: Да, я мечтал об этом со сшколы. Правда, какое-то время я хотел вступить в военно-воздушные силы и летать на самолетах, но вовремя передумал.

Я пошел в Калифорнийский государственный университет Нортридж, и там у меня был роман с девушкой, отец которой работал в НАСА. Она устроила мне собеседование, и вот уже 22 года я здесь. Так началась моя работа в НАСА.

Эндрю: **Расскажите нам немного о ваших проектах, особенно связанных с программированием. Мы бы хотели поговорить о командах в целом и о командах программистов в особенности — о командах, которые хорошо работают вместе.**

Питер: Создать команду бывает нелегко. Порой трудно подобрать людей с определенным набором навыков и личностных характеристик.

Я работал в ЛРД над многими полетами – от «Магеллана» в 1980-х, который картографировал поверхность Венеры, до «Марс Обсервера», который за три дня долетел до Марса, а затем исчез. Я работал над миссией «ТОПЭКС-Посейдон» (океанографический искусственный спутник Земли), а также над системой наземного обеспечения для миссии «Кассини».

«Дип Спейс-1» был моим первым проектом, где я выступал в роли лидера команды: три года я возглавлял группу разработчиков систем контроля полета.

Последние четыре с половиной года я занимаюсь «Фениксом». Я был системным инженером-программистом, ответственным за общую разработку программ как для полета, так и для наземных систем всего проекта. С тех пор я перешел в программу продвинутых оптических систем ЛРД.

Эндрю: **Судя по всему, у вас за эти годы было множество очень важных проектов.**

Питер: Да, действительно много, и о паре я еще не упомянул.

Эндрю: **Вы сказали, что создание команды может быть непростой задачей. Почему? Какие именно проблемы у вас были?**

Питер: Здесь две проблемы. Одна из них – это формирование нужного набора навыков. Нужны люди различных способностей и вкусов. Нельзя брать в команду одних гиков, потому что нужно, чтобы кто-то писал документацию, был системным инженером и занимался тестированием. Кроме того, нужны люди, способные общаться с потребителем.

В любой успешной компании можно найти множество таких людей. Итак, один аспект – это техническая подготовка членов команды.

Второй аспект – это личностные качества. Кто-то более замкнут, кто-то более открыт, и до какой-то степени разобраться в этом сложно. Надо найти правильный баланс личностей, чтобы люди сошлись. Если в команде слишком много сильных личностей, то они могут сцепиться друг с другом.

С другой стороны, если слишком много членов команды – посредственности, то вряд ли они выдадут много блестящих идей. Нужен баланс. И нужна такая среда, чтобы люди могли спокойно делиться своими идеями и проблемами.

Дженни: **Вспомните, пожалуйста, бывало ли так, что в вашей команде все шло хорошо, что вы подобрали идеальный сплав профессиональных навыков и личностных качеств?**

Питер: Думаю, мне в основном везло, так что большинство команд, в которых я работал, были успешны, но некоторые лучше других.

Но в принципе, наверное, команда «Дип Спейс-1» была самой спаянной. Наши программисты составили отличную команду. Думаю, у нас были очень хорошие отношения. Кое-кто из команд интеграции и тестирования порой бодался друг с другом, но все же команда была очень неплохая.

Хорошо работал и «Феникс». Программисты там были очень талантливы и хорошо понимали именно то, что нужно проекту.

Эндрю: **Как вы считаете, всегда ли такое бодание хорошо для проекта или порой дело обстоит иначе?**

Питер: Не уверен, что оно всегда играет положительную роль. Место для дискуссии всегда должно оставаться, но в конце концов люди должны быть готовы принять любое направление проекта.

У нас в ЛРД есть очень, очень умные люди, во многих отношениях гораздо талантливее меня. Но я видел, как такие люди порой заводят в тупик весь коллектив, и выбраться оттуда очень непросто, потому что два или

больше сотрудника просто не хотят признать свою неправоту, что вредит делу.

До какой-то степени нужен здоровый скептицизм, здоровый обмен различными мнениями. Но в конце концов нужно принять одно решение, пойти по одному пути, и всем необходимо быть готовыми этот путь принять и сделать все, что можно, на благо проекта.

Эндрю: **Если я правильно понимаю, то для единства решения нужен лидер, который не только обладает полномочиями, но и пользуется таким уважением в команде, что сможет справиться с разногласиями и принять решение.**

Дженни: **Как вы считаете, помогает ли это команде стать именно командой? Что особенного делаете вы, когда руководите командой?**

Питер: Интересный вопрос.

В работе лидера команды большую роль играет дипломатия, во всяком случае в «Фениксе». Последние четыре года мне пришлось иметь дело с тремя различными организациями – Университетом Аризоны, компанией космических систем «Локхид Мартин» в Денвере и Лабораторией реактивного движения.

Мои разработчики независимо друг от друга создавали собственные программы во всех трех организациях. Не буду упоминать здесь все соглашения, контракты, цепочки руководства и так далее. Дипломатия – это именно то, что нужно в подобной среде.

Чтобы спаять команду, я люблю проводить различные мероприятия, помогающие тимбилдингу. Но в данном случае, когда твоя команда находится в нескольких часовых поясах, это несколько затруднительно. И хотя это прозвучит глупо, но один из приемов, которые я довольно успешно использовал, – это футбольный менеджер.

Я предложил всем вступить в созданную футбольную лигу и привлек к этому с дюжину сотрудников – не так много для всей команды программистов, но они представляли различные подразделения: были люди из операционной команды, из интеграции, тестеры, программисты, системные инженеры – все они были вовлечены в эту игру, а о ней хотя бы можно поговорить, и это делает взаимоотношения более теплыми.

Иначе можно упереться только в чисто профессиональные отношения. И, думаю, тут легко позабыть о том, что перед вами человек со своими чувствами и прочими человеческими качествами.

Это первое, что я стараюсь делать.

Другой метод состоит в том, что я стараюсь быть доступным всем, у кого возникают какие-то вопросы или проблемы. Я придерживаюсь полити-

ки открытых дверей. Любой может прийти и поговорить со мной о своих заботах в любое время. Если проблема серьезная, мы стараемся ее решить. Иногда приходят просто облегчить душу, без последствий.

Думаю, это два самых важных метода в моем арсенале.

Дженни: **А как вы вообще пишете программы? Я знаю, что в НАСА очень много ограничений по процессу. Не расскажете ли, как такой процесс влияет на вашу команду и насколько это мешает вам работать над программами?**

Питер: Ну, насчет ограничений по процессу, не знаю – полагаю, что пилотируемые программы более процессозависимы, чем мы. Честно говоря, за последние 10 лет мы в ЛРД вряд ли так уж сильно зависели от процесса. И надо сказать, что наши партнеры по индустрии далеко ушли от нас в этой области, хотя сейчас, думаю, мы начинаем их догонять.

Но и нам нужно следовать определенным правилам, у нас есть стандарт кодирования, как в любой другой уважающей себя программистской конторе. У нас есть жизненный цикл разработки, которого мы придерживаемся. У нас есть некоторые организационные стандарты и практики.

Прежде всего вам как лидеру команды следует убедиться, что: 1) все осведомлены о действующих стандартах и процессах; 2) что все понимают, как их интерпретировать и применять; 3) что они уже привыкли к тому, что их действия будут проверяться.

Но у меня в основном не было проблем. Люди обычно довольно восприимчивы, по крайней мере те, с кем я работал, и соглашаются с ограничениями данной среды.

Дженни: **Не могли бы вы вкратце описать процесс создания программ?**

Питер: Здесь мне придется соблюдать осторожность, потому что надо помнить об ограничениях ПМТО.

Эндрю: **Что такое ПМТО?**

Питер: Правила международной торговли оружием. Это федеральный закон, запрещающий нам раскрывать информацию, касающуюся оборонных технологий, к которым относятся и космические.

Поэтому ничего подробно говорить не буду, но мы пользуемся стандартным жизненным циклом разработки при каскадном подходе. Иногда разработка итеративна, то есть у вас есть серия конструкций, которые вы собираете и встраиваете функциональность, двигаясь к конечной цели – полнофункциональной системе.

У вас есть требования к разработке, дизайн, этапы интеграции. Здесь мы с этим хорошо знакомы. Когда появляются новички, они быстро разбираются.

Эндрю: **Тут я почти что разочарован: оказывается, вы тоже смертные. А я всегда представлял себе, что у НАСА есть какие-то невероятные, хитрые способы создания ПО. Но, похоже, вы делаете то же самое, что и остальные, просто вам приходится быть намного осторожнее и более четко придерживаться правил, чем всем остальным. Думаю, здесь дело в том, что ваше оборудование должно работать в миллионах миль отсюда, куда вы уже никак не можете забраться.**

Питер: Да, это так.

Эндрю: **И как вы с этим справляетесь?**

Питер: Мы делаем много тестов. Мы постоянно занимаемся тестированием, для этого есть множество системных инженеров. У нас комплексная команда. Многие работают и в отделе системной инженерии для космического корабля или миссии, и в отделе программирования.

Концы с концами обычно сходятся в области защиты от ошибок. У нас есть команды из двух-четырех системных инженеров, которые рассматривают систему целиком, выясняют, что с ней может быть не так и как можно это исправить. Они входят и в команду программистов, помогая писать спецификации для программ, осуществляющих наладку.

Здесь-то все и сходится. Защита от ошибок и работа над ними – это то, что позволяет нашим миссиям работать длительное время без прямого взаимодействия с операторами.

Эндрю: **Думаю, не каждый, кто прочтет это интервью, знаком с принципами защиты от ошибок и вообще тестированием. Нельзя ли, не нарушая никаких федеральных законов, немного рассказать нам о том, как вы тестируете подобные программы?**

Питер: Да, могу дать общий обзор. Здесь важно прежде всего представлять себе, что должна делать эта программа, чтобы впоследствии не обнаружились какие-то скрытые команды, данные, функции.

Все должно быть доступно, чтобы вы видели все, что происходит в программе, как можно более ясно. Разумеется, не надо публиковать каждый регистр, или переменную цикла, или промежуточное состояние. Но по крайней мере все аппаратные измерения должны быть доступны для пользователя.

Второе, что здесь важно, – наличие платформ для тестирования. В последние десять лет мы работаем в основном с программным симулятором.

У вас есть программный симулятор, который может запускаться на рабочей станции и моделировать в общих чертах ту среду, в которой функционирует космическое программное обеспечение. У вас есть модели систем авиационной электроники. Есть модели механической динамики – крутящего и поступательного моментов. Модели планет и всех остальных внешних факторов, которые воздействуют на вашу систему.

Итак, вы можете запускать программы в среде, которая симулирует условия их реального функционирования. Вы проводите тестирование и подтверждаете, что поведение именно таково, как вы и ожидали, как в расчетном, так и нерасчетном режиме, так что можно выявить случаи, в которых защита от ошибок работает успешно, а также случаи, когда ее нужно каким-то образом улучшить.

Мы пользуемся и аппаратными испытательными стендами. Это самый надежный испытательный стенд. Вы создаете копии электроники, помещаете их в испытательный стенд и управляете ими в условиях, приближенных к полету. Вы наблюдаете за их поведением и надеетесь, что оно именно таково, как вы и ожидали.

Это лучший вариант, но вы ограничены тем фактом, что обычно аппараты функционируют в реальном времени. Тут есть две проблемы – времени и денег: дополнительный аппаратный набор очень дорог. Поэтому можно провести всего пару аппаратных стендовых тестов, и не более. Можно положиться на программные испытательные модели, поскольку они обычно работают быстрее, чем в реальном времени, и их можно дублировать на неограниченном количестве рабочих станций.

Эндрю: **Прежде всего, спасибо за объяснения, это очень интересно. Мне кажется, что если бы вы, например, работали в типичной команде программистов, которая создавала бы типичные бизнес-приложения в какой-нибудь компании, и заставляли бы их заниматься такого рода тестированием на высочайшем уровне контроля качества, в команде начались бы брожения.**

Питер: Вот поэтому наши миссии на Марс стоят 400 миллионов долларов, а ваше типичное бизнес-приложение – всего миллион. Посмотрите на разницу и подумайте о масштабах и цене ошибки.

Цена ошибки в том случае, если бизнес-приложение не будет работать корректно, – всего лишь несколько разгневанных покупателей. Для бизнеса это может быть плохо, особенно если утихомирить их не удастся, но предполагается, что в расчетном режиме программа будет работать хорошо, а в нерасчетном команда получит жалобы и выпустит патч.

Если же у нас в нерасчетном режиме что-то пойдет не так, то второго шанса мы не получим..

Эндрю: **Предполагаю, что, в отличие от бизнес-команд, члены вашего коллектива, даже самые молодые, более сосредоточены на качестве, больше ценят качество. А возможно, и весь свой проект.**

Питер: Да, думаю, те, кто работают с нами, обычно считают это привилегией. Я так точно считаю. Это уникальное место работы. Конечно, у него есть кое-какие недостатки, но есть и множество преимуществ, в том числе сама возможность работать над тем, чем мы занимаемся здесь.

Думаю, что большинство сознает это и старается, как может. Они понимают, что наши программы трудоемки и требуют внимания к деталям.

И поскольку мы все-таки можем что-то пропустить, потому что все мы люди, у нас есть суровая тестовая программа, которая призвана вскрыть все крупные недостатки. Мы не стремимся устранить абсолютно все дефекты, но если исключить наиболее крупные, то все будет нормально.

Цель в том, чтобы сделать космический корабль устойчивым, так чтобы ничто из сделанного нами не смогло ему повредить. Это основная задача: корабль должен быть настолько жизнеспособным, чтобы самостоятельно защитить себя, пока мы не разберемся, в чем дело, и не исправим ошибку.

Эндрю: **Предполагаю, что все это внимание к деталям, к качеству, осознание необходимости правильности своих действий и чувство привилегированности должны позитивно влиять на команды и их способность быстро срабатываться и двигаться к одной цели.**

Дженни: **У меня другой вопрос. Одна из проблем, которые мы обсуждаем с нашими гостями, состоит в том, что часто цели у членов команды различны. Они понимают задачи проекта несколько иначе, чем другие.**

Встречались ли вы с подобным в ЛРД или для вас это не актуально? Мол, наш космический корабль летит на Марс, и это цель нашего проекта.

Питер: Да, вроде бы во всех проектах, связанных с полетами, над которыми я работал, ребята очень хорошо ориентированы на цель и на то, к чему должен прийти данный проект.

Обычно бывает, что кто-то обязательно двигается не совсем в том направлении и делает не совсем то, что нужно, но большинство сотрудников понимает главную цель и осознает серьезность того, что может случиться, если у нас не получится достичь ее с первого раза.

Эндрю: **Влияет ли это на то, как люди работают вместе?**

Питер: Думаю, на повседневные действия команды в основном влияют, во-первых, характеры людей, как я уже говорил, а во-вторых – не забы-

вайте, наш путь не усеян розами. Быть здесь – честь и привилегия, но не надо забывать и о грузе ответственности.

Я умею расслабляться, но мне известны люди, на которых действительно давит сознание важности того, что они делают.

Многие месяцы мы работаем сверхурочно. Нужно успеть запуститься во время астрономических окон, которые длятся три недели и бывают примерно раз в два года. Пропустить время запуска нельзя.

Если не успеть к этому времени, это приведет к потерям десятков, а то и сотен миллионов долларов, потому что вместе со всем оборудованием и командой придется ждать новых благоприятных сроков. Нужно будет хранить технику, доводить до ума дополнительные возможности – в общем, платить придется за все. Мы действительно очень стараемся уложиться в эти сроки.

Отсюда множество стрессов, и известны случаи личных проблем, ставших результатом рабочего расписания, которого мы порой вынуждены неукоснительно придерживаться.

Дженни: **Итак, с учетом подобных стрессов и суперстрогих дедлайнов, считаете ли вы, что это мешает общению в команде? Или это, наоборот, объединяет?**

Питер: Думаю, тут многое зависит от человека и от того, укладываетесь ли вы в расписание.

Честно говоря, те, кто не может выдержать стресс, обычно просто покидают команду. Они уходят на другую работу или переходят куда-то внутри нашей компании. В этом бизнесе не место тем, кто не может справиться с работой. С учетом этого, думаю, можно говорить о хорошем сотрудничестве и общем взгляде на принципы работы.

Что до работы в команде, то обычно всем командам она удается. В тех, где работал я, особых разногласий не было. Иногда были разные мнения по поводу того, что сделать в первую очередь, но обычно, если направление задано, то дальше все идет как по маслу.

Где у нас проблемы, так это, думаю, с повторным использованием. Было несколько попыток создать повторно используемое ПО, чтобы снизить общую стоимость проекта и сделать проекты в целом более окупаемыми.

Честно говоря, думаю, что отсутствие в этом случае четкого дедлайна привело к тому, что члены команды расходились во мнениях по поводу того, в каком направлении двигаться, и в итоге этот проект не получил должного развития.

Дженни: **Очень интересно. Другие наши гости, у которых мы брали интервью для этой книги, тоже выдвигали идею о том, что само по**

себе повторное использование как цель порой может стать скорее препятствием.

Питер: Да, думаю, это утверждение соответствует действительности.

Дженни: **Для меня самое интересное здесь то, что многие считают повторно используемое ПО целью множества проектов. Я работала во многих командах — и слышала о таких, которые целые программные пакеты создавали с учетом возможности повторного использования, особенно за последние пять-шесть лет. Вероятно, это характерно для всей отрасли.**

На практике, при вашем строгом подходе к делу — есть ли у вас проблемы при создании ПО, подразумевающего несколько уровней абстракции, если цель — повторное использование?

Питер: Думаю, да. Мы пытаемся делать это вот уже 20 лет. Но честно говоря, я все еще не уверен, что мы делаем это правильно.

Вот как обычно бывает: вы собираетесь построить какую-то повторно используемую платформу, не обязательно ПО. Это относится и к аппаратному обеспечению, и к авиационной электронике, и к другим компонентам, которые мы используем.

Вы собираете вокруг себя умных людей, но между ними на самом деле нет согласия по поводу того, как лучше это сделать, и вот они начинают спорить, разговаривать об этом и достигают некоторого успеха. Но вот возникает какой-то проект, который может или должен использовать эту систему, и внезапно система оказывается полностью подчинена этому проекту.

Вот так, добившись успеха в создании настоящей функциональной системы, вы теряете возможность ее повторного использования, потому что теперь она полностью подчинена проекту, и деньги этого проекта нельзя тратить на благо следующего. Деньги проекта направлены на то, чтобы запустить его и сделать так, чтобы он функционировал наилучшим образом.

Это крепкий орешек. Честно говоря, не знаю в ЛРД никого, кто бы преуспел в создании повторно используемой платформы.

Эндрю: **Может быть, как только общая ясная цель вывода космического корабля на планетарную орбиту достигнута, цель становится более расплывчатой? Считаете ли вы, что отсутствие четкой цели может сделать команду менее сплоченной?**

Питер: Да. Начинаются философские споры, уход в абстракцию — и хотя вы думаете, что перешли на более высокий уровень, но именно

на нем становится тяжело объяснить, почему данное конкретное направление работы является правильным.

Когда вы ставите конкретную задачу: «Нужно сделать вот такую функциональность, вот к такой дате, а то мы пропустим астрономическое окно», – обычно с вами соглашаются. Люди говорят: «Ладно, это не совсем то, чего бы мне хотелось, но работать будет, и это хорошо».

Собираясь работать над повторно используемым ПО, вы теряете это отношение «и это хорошо», это естественно, потому что вы пытаетесь найти наилучшее решение, а не просто хорошее. Но это, как я уже сказал, может стать препятствием к достижению цели.

Эндрю: **Похоже, здесь возникает угроза бодания, о котором вы говорили, только в этом случае уже нельзя сказать что-то вроде: «Нужно сдать миссию. Окно запуска длится всего три недели, и если мы будем друг другу что-то доказывать все следующие шесть месяцев, то пропустим его». Этого козыря больше нет, и ситуация становится тупиковой.**

Питер: Именно так. Еще одна проблема в ЛРД состоит в том, что каждая наша миссия уникальна – кстати, сама эта проблема, видимо, тоже уникальна. Я еще не встречал миссий, которые были бы идентичны или просто очень похожи.

Посмотрите, например, на Mars Pathfinder, MER и Mars Science Laboratory – последний проект сейчас в разработке. Все это марсоходы. Однако каждая миссия уникальна по бортовому аппаратному обеспечению, методам доставки их на Марс и используемым пакетам инструментов.

Сейчас мы имеем дело с ситуацией, когда технология развивается быстрее, чем жизненный цикл. От идеи до запуска миссии проходит от трех до пяти лет. Технология развивается примерно теми же темпами, даже быстрее: посмотрите, например, как быстро устаревают компьютеры.

И мы остаемся в положении, когда даже если мы согласны разрабатывать определенную платформу на определенном языке и определенном аппаратном обеспечении, то технология к концу подготовки миссии все равно шагнет вперед. И работа становится еще сложнее, чем просто создание стандартной платформы многоразового использования.

Мы по-прежнему работаем над этим, и многие умные люди все еще обдумывают и эту идею, пытаясь прикинуть, что делать. Возможно, когда-нибудь с помощью подключаемых архитектур мы придем к чему-то более универсальному.

Дженни: **У меня возник вопрос, еще когда вы говорили о команде тестеров. Один из главных принципов Agile-разработки, который так**

нравится нам с Эндрю, заключается в том, что ответственность за качество ложится на самих разработчиков, что они обязаны тестировать собственный код, а непрерывная интеграция облегчает отслеживание внесенных в код изменений. Применяете ли вы у себя что-то из этого?

Питер: Не думаю, что у нас кто-то применяет Agile-разработку.

Дженни: **Не Agile-разработку как таковую, а методы контроля качества, применяемые самими разработчиками.**

Питер: Да, мы требуем от наших разработчиков осуществлять модульное тестирование кода перед представлением его для интеграции.

Обычно мы не занимаемся непрерывной интеграцией. Чаще всего мы поступаем так: все собираются вместе, мы вырабатываем требования и дизайн для текущего жизненного цикла, делаем модули, тестируем их и тут же сращиваем.

Эндрю: **У меня вопрос об интеграционном тестировании. То, что вы сейчас сказали, напомнило мне, как я впервые узнал об интеграции, еще когда только учился программированию.**

Если вы не изобрели термин «интеграция» по отношению к программированию, то точно популяризировали его в 1980-е – начале 1990-х. Для нас, работающих только с программами, каждая система проекта – это программная система, а интеграция – это просто сведение модулей. Но я полагаю, что, говоря об интеграции, вы имеете в виду более широкое и старое значение этого слова: интеграцию разработки программного и аппаратного обеспечния. Возвращаясь к этому значению, мы видим, что и модуль – как в словосочетании «модульное тестирование» – тоже имеет иной смысл. Итак, что же вы понимаете под интеграцией? Интеграцию программного и аппаратного компонентов внутри более крупной системы, над которой вы работаете?

Питер: Да. Итак, есть два уровня интеграции. Мы разрабатываем независимые программные компоненты, или модули, которые достаточно малы, чтобы один-два человека могли поднять их, адекватно подключить и протестировать.

Затем дело доходит до нашей программной интеграции. Мы объединяем модули, компонуем их и запускаем тестирование программной интеграции, например на одной из тестовых платформ SoftSim, которые я уже упоминал, или даже на аппаратном тестовом стенде, если он нам доступен.

Когда ПО интегрировано и готово к работе, менеджер программ объявляет его готовым, и оно отправляется на тестовые стенды для сертификации. Затем оно направляется на механизм.

Один из этапов миссии называется ОКИЗ (Операции по комплектному испытанию и запуску). Он также известен как Фаза D (просто буквенное обозначение этого периода в жизненном цикле проекта). Она начинается с поставки аппаратуры на интеграцию – в так называемый заводской цех, хотя мы стараемся заниматься этим в чистовой комнате.

Начинается сведение аппаратуры и ее сборка. Когда достигнута достаточная критическая масса собранной аппаратуры, можно загрузить ПО в систему авиационной электроники, начать тестирование, чтобы убедиться, что: 1) ПО правильно управляет аппаратурой; 2) аппаратура правильно ведет себя под управлением ПО.

Мы тестируем так много, как только можем. Обычно перед запуском этап тестирования – ОКИЗ – занимает от 9 до 12 месяцев. Когда новая программная сборка готова, она отправляется на аппаратный тестовый стенд, а затем на ОКИЗ для интеграции с механизмом и нового тестирования. Таким образом, у нас несколько уровней интеграции.

Эндрю: **Пожалуй, те, кто узнав, что вы не занимаетесь непрерывной интеграцией, недоумевали по этому поводу, получили исчерпывающее разъяснение. Думаю, вполне можно сказать, что если ваш процесс интеграции включает чистовую комнату, то он исключает автоматическую интеграцию.**

Питер: Верно. У нас нет автоматической интеграции. Я видел у некоторых наших партнеров по индустрии системы с автоматизацией: например ночная сборка всего, что было за день проверено вручную. Смысл здесь в том, чтобы убедиться, что всей системе не повредили какие-либо внесенные изменения.

Затем в определенный момент – или когда все за – мы решаем, что данная сборка готова для интеграции, и процесс движется дальше.

Насколько я понимаю непрерывную интеграцию, в вашем случае любой может сказать: «Вот мой модуль готов», – и начать с ним работать. Мы же этого сделать не можем. Наши усилия требуют более существенного координирования.

Эндрю: **Одно из преимуществ непрерывной интеграции или автоматического интеграционного процесса – его эффективность, обеспечивающая достаточный уровень качества. Не тратя много времени на координирование при создании программ, можно при этом не сомневаться в надлежащем уровне качества программы: можно положиться на то, что система работает чисто, проходит модульные тесты и так далее. Но автоматизации вы не можете так довериться – по вышеизложенным причинам. Есть ли другие способы повысить эффективность вашего процесса? Способы убедиться, что качество вашего кода всегда высоко?**

Питер: Иногда мы запускаем эти циклы параллельно, то есть сборка номер два будет в интеграции, а сборка три – на этапе проектирования. Сотрудники одновременно работают над разными аспектами системы на разных этапах жизненного цикла.

Да, мы надеемся, что наши сотрудники выдают качественный код, прошедший модульное тестирование. И не только: он должен удовлетворять стандартам кодирования. Он должен иметь подробные комментарии и сопровождаться документацией, которая описывает его рабочие свойства, сообщения об ошибках, обнаруженных при последней проверке, и об ошибках, которые до сих пор не устранены. Мы продолжаем отслеживать все дефекты, найденные при интеграции и тестировании, и должны обнаружить их все до запуска.

В НАСА есть специальное подразделение – отдел независимого контроля и утверждения. Он находится в Фэрмонте (Западная Виргиния), и проводит независимый аудит наших процессов и кодов, помогает выявить оставшиеся вопросы, ошибки, дефекты, требования.

Дженни: **Как независимая группа обеспечения качества ПО?**

Питер: Да, именно как внешняя SQA-группа[1]. У нас в ЛРД есть независимые SQA-инженеры, которые подчинены не менеджерам проекта, а нашей главной SQA-организации. Помимо всего этого, у нас есть внешняя SQA-группа, которую предоставляет НАСА.

Дженни: **Вроде бы все очень просто. Программисты пишут код и документацию по нему. Они проходят через множество проверок – как самостоятельных, так и коллективных?**

Питер: Да.

Дженни: **Отлично, если требуется выявить дефекты как можно раньше. У меня вопрос: за годы работы я заметила, что многие разработчики сразу принимают защитную стойку, когда другие смотрят их код, нелестно комментируют его и даже находят ошибки. Как вы считаете, у вас в команде другое отношение?**

Питер: Думаю, да. Разработчики, с которыми я работал, были достаточно ответственны, чтобы понять, что их код необходимо проверять на предмет выявления возможных проблем. Собственно говоря, мы проверяем код каждого модуля, который предназначен для космического корабля.

Конечно, приходится считаться и с профессиональной гордостью. Никому не нравится, когда твой код критикуют, но с другой стороны,

[1] От Sofrware Quality Assurance – обеспечение качества ПО.

я никогда не сталкивался с тем, чтобы кто-то не признавал наличия проблемы и отказывался ее решать. Во многих случаях каждый был только рад, когда ему говорили: «Смотри, тут ошибка параллельной обработки, здесь возможно зависание». Он отвечал: «Точно, так и есть. Надо исправить». Никогда у меня не было с этим особых проблем.

Дженни: **А как насчет взаимоотношений между тестерами и разработчиками? Они хорошие?**

Питер: В основном да. Иногда, впрочем, это вообще один и тот же человек. Мы не всегда проводим независимое тестирование. У нас есть независимые проверки результатов тестов. То есть свои модульные тесты разработчики делают сами. Потом идет их проверка, а когда доходит дело до интеграции, то часто сам разработчик пишет интеграционные тесты, а иногда их и выполняет.

Но у нас всегда есть другая пара глаз – кто-нибудь из системного или подсистемного отдела, отвечающий за ту часть оборудования, которой должна управлять данная программа.

Системный инженер обязательно изучает тесты и определяет, что программа, судя по всему, функционирует правильно.

Эндрю: **Да, просто поразительно, насколько вы заботитесь о качестве, и результаты миссий говорят сами за себя.**

Но могу представить, как наш читатель, особенно с опытом разработки «вслепую» и выдачей продукта на-гора, часто не в самой удачной обстановке, получив по внутренней почте письмо от службы независимой проверки или группы SQA, открывает его и стонет: да что ж такое, опять какой-то отчет какого-то деятеля, я его и знать не знаю, а он просмотрел весь мой код и нашел кучу ошибок!..

Питер: Что ж, это не пикник. В подобных условиях действительно нелегко.

За время «Феникса» служба независимой проверки НАСА прислала больше 800 технических записок о проблемах. И каждую из них мне пришлось просмотреть, чтобы убедиться, что эта проблема не критична для жизнеспособности космического корабля.

Где-то с полдюжины из них действительно были значительными, так что пришлось их устранять. Многие из них были отловлены еще на внутренних проверках или тестировании. Некоторые напоминали блошиные укусы, о них можно было сказать: «Да-да, я понял, о чем вы говорите, но это не должно вызвать проблему: мы либо используем это по-другому, и ошибки не будет, либо это вообще несущественно; да,

с точки зрения стиля то, что вы предлагаете, лучше, но и наш вариант вполне приемлем».

Эндрю: **Как вы считаете, если попросить всех членов команды честно высказаться, то каким будет общее мнение: стоит или не стоит все это усилий? Мог ли космический корабль из-за этих пяти-шести ошибок действительно врезаться в Луну?**

Питер: Ни одна из этих ошибок не была критичной для миссии. Даже если бы мы их не устранили, миссия не была бы завалена. Но они могли быть неприятными. Пришлось бы провести на Земле лишнюю пару дней или даже неделю, выясняя, что случилось и что теперь с этим делать.

Хорошо, что мы их обнаружили. Стоило ли это усилий? Думаю, да. Определенно, стоило. Это как есть овощи: не всегда хочется, но зато полезно.

Эндрю: **По-моему, очень интересно. Если распределить методики создания программ и контроля качества по шкале от самых легких до самых трудных, то вы будете в самом конце. Но вы сами, кто должен был анализировать все эти записки, до сих пор считаете, что это оправданно. Это стоило потраченных вами и вашей командой времени и сил: готовый продукт в результате стал лучше.**

Питер: Да, я так думаю. В случае с «Фениксом» у нас было очень мало проблем в полете. До посадки меня не беспокоили ни по каким серьезным вопросам, связанным с программным обеспечением.

А после посадки обнаружилась одна проблема, связанная с повторным использованием системы телеметрии, изначально созданной для орбитальной ступени, где не ожидалось, что ее будут часто перезагружать.

А теперь ее выключали и включали по нескольку раз в день, и один счетчик никак не управлялся нормально. Из-за этого счетчика возникли проблемы с переходом космического корабля в так называемый безопасный режим, когда корабль сворачивает все научные наблюдения, пытается поддерживать приемлемый энергорежим и связывается с Землей в ожидании инструкций.

К счастью, источник проблемы удалось обнаружить довольно быстро, в течение 24 часов мы поняли, в чем дело, и через несколько дней уже все исправили.

Но за все время моей работы с космическими миссиями именно с этой было меньше всего программных проблем.

Дженни: **И последний общий вопрос, Питер. Назовите конкретную черту, которая характеризовала вашу лучшую команду. Найдется у вас перл мудрости для наших читателей?**

Питер: Думаю, лучшие команды, в которых и с которыми я работал, отличались сознанием общей цели. Вы работаете для одной цели и цените друг друга. Вы уважаете навыки друг друга, и хотя один из вас – лидер проекта, а другой – главный специалист по архитектуре, все понимают, что вклад в дело общий.

Как ни странно, наибольшее удовлетворение не всегда приносят проекты, связанные с чем-то величественным, но те, где команда сплоченная и все друг друга уважают за работу.

19

Успешное техническое задание

Драма в трех актах

Карл Е. Вигерс

Время и место действия

Осень 1994 года, исследовательские лаборатории крупной компании Contoso Pharmaceuticals (компания настоящая, название вымышленное).

Действующие лица

Пол

Менеджер отдела охраны труда и промышленной безопасности Contoso Pharmaceuticals

Дана

Менеджер склада химреагентов Contoso

Сара

Химик-исследователь, исследовательские лаборатории Contoso

Джонатан

Сотрудник отдела закупок

Джанет

Менеджер проекта и аналитик ИТ-отдела исследовательских лабораторий

Девон

Программист-аналитик ИТ-отдела исследовательских лабораторий

Карл

Консультант исследовательских лабораторий Contoso

Пролог: Пол попадает в переплет

У Пола возникли проблемы. Государственные органы ввели новые положения, по которым Contoso Pharmaceuticals должна предоставлять ежеквартальную отчетность по приобретению, хранению, использованию и утилизации химреагентов. В лабораториях Contoso сотни химиков и обширные запасы реагентов, тысячи дополнительных контейнеров хранятся на складе. Единственный способ выполнить требования по отчетности для Пола – это организовать надежную Систему учета химреагентов, которая позволит отслеживать местонахождение и статус каждого контейнера с химреагентом в компании.

Пол уже довольно давно знает об этих требованиях, но сейчас дело дошло до критической черты. Уже две команды из ИТ-отдела компании пытались создать такую систему учета. Каждая работала с Полом, обсуждала требования, но ни одна так и не написала техническое задание, и, в конце концов, оба начинания сошли на нет. Зная о важности и необходимости такой системы, отдел информационных технологий собрал новую команду, чтобы наконец выполнить эту задачу.

Акт I: закатав рукава

Мы разрабатываем стратегию и набираем людей.

Первоначальная цель команды по проекту заключается в разработке достаточно четкого и полного технического задания, чтобы можно было приступить к разработке ПО. Решение о том, будет ли система проектироваться в компании или эту задачу частично либо полностью отдадут в аутсорсинг, отложено до более поздних сроков. Наша команда по сбору и анализу требований состоит из трех человек:

- Джанет – опытный разработчик и аналитик, впервые берущий на себя руководящую роль в серьезном проекте. Ей предстоит планировать деятельность по проекту, отслеживать ход работ и параллельно выполнять работу аналитика. Джанет, хоть и молода, имеет зрелый ум. Ее не просто сбить с толку, характер спокойный.
- Девон – менее опытный разработчик. Талантлив и энергичен, мечтает приобрести навыки аналитической работы. Если система будет разрабатываться в компании, Девон будет в этом участвовать.

- Я, Карл. Я работаю в Contoso уже 15 лет: научным сотрудником, разработчиком ПО, менеджером по программному обеспечению и инженером по качеству. Поскольку я какое-то время работал над улучшением процесса разработки требований и технических заданий, то часто выступаю в роли консультанта в рамках компании. В проекте системы учета я буду аналитиком на полставки.

Джанет встречает Пола и говорит ему, что наша команда аналитиков сделает все возможное для выполнения его требований. Но в первую очередь нам, разумеется, нужно знать его требования к системе учета. Пол уже откровенно злится: «Я дал все требования ребятам из двух предыдущих команд, – говорит он. – У меня больше нет времени болтать о требованиях. Дайте мне систему!» И это становится нашей мантрой: «Дайте мне систему!»

При встрече с враждебно настроенным, разочарованным, даже разозленным главным заказчиком становится понятна сложность нашей задачи. Нам предстоит справиться с неприятным наследством наших предшественников и каким-то образом привлечь Пола и других будущих пользователей к участию в эффективном совместном процессе разработки требований. Похожа, битва будет жаркой.

Процесс разработки технического задания часто приходится вести в непростых, а то и боевых условиях. Участники иногда забывают, что они по одну сторону баррикад (по крайней мере, должны быть), работают на общую цель – успешную разработку полнофункционального продукта. Нашим первым шагом должно стать установление партнерских отношений с Полом. Нужно, чтобы он поверил, что в этот раз все будет по-другому, что мы действительно поможем ему выполнить нормативные требования. И мы, технические специалисты, должны разработать план того, каким образом мы сможем оправдать его ожидания.

Управление ожиданиями

Обычно команда начинает работать над проектом, не договорившись заранее, каким образом будет осуществляться сотрудничество. У членов команды есть свои мнения о том, какую работу нужно делать и как люди будут взаимодействовать друг с другом. У каждого свой стиль общения, свое понимание проблем, свое восприятие даже самого понятия «требования» и так далее. Многие группы не приходят к четкой и однозначной договоренности о том, как они будут принимать тысячи решений, необходимых по ходу проекта. Пренебрежение такими вещами может привести к несоответствию ожиданиям, неэффективному взаимодействию и неприятным моментам. Стоит потратить некоторое время в самом начале на обсуждение того, как команда будет работать.

Джанет начинает с обещания Полу о том, что наша аналитическая команда подойдет к вопросу определения его требований к приложению более эффективно. Более того, мы создадим необходимую и достаточную документацию на основе наших выводов, чтобы к работе смогли приступить разработчики. Джанет также обрисовывает Полу, какая помощь нам потребуется с его стороны, чтобы эта совместная работа могла увенчаться успехом. Вне зависимости от прошлого печального опыта нам придется начать все заново и первым делом выяснить требования. Поэтому нам потребуется время Пола и его коллег, чтобы лучше понять, какая система им нужна. Пол с большой неохотой соглашается участвовать.

Классы пользователей

Не один Пол заинтересован в разработке системы учета, и не только работники Департамента охраны труда и промышленной безопасности будут ее использовать. Мы применяем понятие *классы пользователей*, чтобы определить другие группы пользователей, у которых будут свои внушительные наборы требований. Представителям разных классов пользователей могут потребоваться разные функции и характеристики, у них могут быть разные уровни образования и опыта работы, они могут работать в разных помещениях и географических точках, возможны и другие отличия в характеристиках.

На основании сведений, полученных у Пола, и понимания ситуации в исследовательских подразделениях Contoso, сложившегося у команды аналитиков, мы определяем четыре основных класса пользователей:

Сотрудники отдела охраны труда и промышленной безопасности

Эти сотрудники под руководством Пола отвечают за создание отчетов по работе Contoso Pharmaceuticals с химреагентами в соответствии с нормативными требованиями.

Химики

В исследовательских лабораториях, отделах разработки и производства товаров Contoso работают сотни химиков. Эти люди получают новые химреагенты, хранят их в своих лабораториях, используют для опытов и уничтожают/утилизируют ненужные остатки.

Сотрудники склада химреагентов

Эти несколько человек играют ключевую роль в учете химреагентов. Они размещают заказы на закупку химреагентов у поставщиков, размещают и выдают тысячи емкостей с химреагентами, управляют запасами новых химреагентов, созданных самими исследователями, и уничтожают/утилизируют просроченные химреагенты.

Сотрудники отдела снабжения

Как и персонал отдела охраны труда и промышленной безопасности, эти люди никогда напрямую не работают с химреагентами. Они служат соединяющим звеном между сотрудниками Contoso, размещающими заказ на химреагенты, и поставщиками.

У этих классов пользователей есть некоторые общие потребности. Например, и химики и сотрудники склада химреагентов периодически размещают заказы на новые и уничтожают/утилизируют ненужные химреагенты. Однако у каждого класса пользователей есть свой четкий набор требований к системе учета химреагентов.

Следующая задача нашей аналитической работы – найти конкретных сотрудников, которые могут обрисовать для нас эти требования. Потом на их основе будет сформировано техническое задание на разработку программного обеспечения. Частью работы по определению требований станет разрешение конфликтов требований и расстановка приоритетов, с тем чтобы оптимально быстро и с наименьшими затратами создать функциональную и удобную систему.

К кому пойдем?

В 1986 году моя небольшая группа разработчиков в Contoso осознала, что надо учиться тесно взаимодействовать с нашими пользователями, чтобы суметь более точно уложиться в их требования к ПО. Мы решили ввести в проект роль *сторонника продукта* (product champion), главного представителя пользователей, работающего в тесном сотрудничестве с аналитиками требований. Сторонник продукта должен быть, в буквальном смысле, голосом потребителя из определенного класса пользователей. Кроме того, предполагается, что эти люди являются опытными пользователями и экспертами в своей области.

Таким образом, следующим шагом по проекту системы учета стало определение сторонников для всех четырех классов пользователей. Наши предшественники составили список функций сторонника продукта, который группа аналитиков могла бы предложить кандидатам на эту роль, чтобы они четко понимали, на что идут. Достижение договоренности по ответственности, которую готов взять на себя каждый сторонник, – важная часть работы по созданию необходимых партнерских отношений при разработке с участием пользователей.

Кандидатура Пола как ключевого ответственного руководителя и прямого пользователя будущей системы на роль сторонника продукта от департамента по охране труда и промышленной безопасности представляется очевидной. Менеджер нашего проекта и ведущий аналитик Джейн будет работать с Полом, чтобы понять требования с его стороны.

Сотрудник отдела снабжения Джонатан готов представлять требования к системе учета от имени своих коллег. Наш второй аналитик, Девон, берется работать с Джонатаном и выяснить требования снабженцев. Дана руководит складом химреагентов. Она, таким образом, является «естественным» представителем пользователей этого класса, хотя ей предстоит собрать мнения своих сотрудников, ежедневно работающих с товаром. Джанет берет на себя дополнительную роль по выяснению требований этих людей.

Гораздо сложнее выделить сторонника продукта в группе химиков. Дана пытается помочь нам. «Прежде чем возглавить склад, я работала химиком в лаборатории, – говорит она. – То есть вам не придется говорить с другими химиками и выяснять их потребности, я могу рассказать вам все, что вам нужно знать».

И хотя Дана предлагает это от души и из лучших побуждений, она не вполне права. Только очень непросто убедить ее в неправоте. Во-первых, она уже не представляет этот класс пользователей – химиков. Она, несомненно, лучше всех на свете может представить потребности сотрудников склада химреагентов. Однако она уже не работает химиком. Издавна я усвоил, что в процессе разработки требований лучше всего задействовать тех, кто фактически представляет конкретный класс пользователей на данный момент, а не искать альтернативу, призывая на помощь бывших работников. В некоторых случаях, особенно в проектах разработки продуктов для рынка, может не быть возможности напрямую работать с представителями соответствующего класса пользователей, поэтому приходится обращаться к «заменителям», менеджерам по этому продукту или маркетологам. Но в нашем случае внутреннего проекта такой проблемы нет, и мы можем организовать работу с фактическими будущими пользователями.

Второе наше опасение, связанное с предложением Даны, заключается в том, что у нее очень узкое и ограниченное понимание требований химиков. Кроме того, она всегда непоколебима в своем мнении по данным вопросам, и когда спор становится жарким или ее довод не сразу принимается, она быстро срывается на крик. (Работать с Даной не самое большое удовольствие.) Полагаться только на безапелляционные мнения Даны не лучший способ глубоко и полно понять все требования химиков к системе. Поэтому мы вежливо отклоняем ее предложение и отправляемся на поиски подходящего представителя от химиков.

И находим его! Очень опытный и уважаемый химик Сара предлагает выступить сторонником продукта. И хотя она не очень хорошо разбирается в компьютерах, Сара признает пользу, которую система учета химреагентов может принести ей и ее коллегам. Часто те, кто лучше всего подходит на роль представителя от своего класса, уже и так перегру-

жены работой и неохотно отрывают время от нее на то, чтобы помогать в разработке требований. Однако нам очень повезло с Сарой. Она обещает найти время в своем плотном расписании, чтобы участвовать в обсуждениях требований и проверке результатов. Поскольку я по образованию химик, мне и быть тем аналитиком, который выяснит у Сары требования, предъявляемые химиками к системе учета химреагентов.

Мы понимаем, что невозможно ожидать даже от такого опытного химика, как Сара, знания всего спектра требований по всем химикам в Contoso. Ей понадобится помощь. И мы собираем группу поддержки, пять химиков из разных отделов компании. Многие их требования совпадут, но этим людям известны и конкретные потребности химических исследований, проводимых в той области, которую они представляют.

Мы не ждем, что Сара в одиночку или даже при поддержке этой группы составит требования по всему классу химиков. Каждый из четырех сторонников продукта отвечает за взаимодействие с другими пользователями своего класса, сбор идей и откликов на предложения. Также предполагается, что сторонники будут разрешать конфликты требований между пользователями одного класса, что значительно облегчит жизнь аналитикам. Таким образом, в этой представительной системе основной поток требований поступает от сторонников продукта к соответствующему аналитику, тогда как за кулисами происходит активное взаимодействие между сторонником и другими пользователями. Аналитик должен работать со сторонниками и разрешать конфликты, возникающие между пользовательскими классами.

И вот мы готовы начать. В табл. 19.1 показан полный список основных участников, задействованных в процессе сбора требований, который мы готовимся запустить. Но с чего именно нужно начинать?

Таблица 19.1. Участники процесса сбора требований для проекта системы учета химреагентов

Класс пользователей	**Аналитик требований**	**Сторонник продукта**	**Группа поддержки**
Сотрудники отдела охраны труда и промышленной безопасности	Джанет	Пол	Нет
Сотрудники склада химреагентов	Джанет	Дана	Нет
Химики	Карл	Сара	Да (5)
Сотрудники отдела снабжения	Девон	Джонатан	Нет

Акт II: варианты использования

Мы пробуем разные методы разработки требований и смотрим, что получается.

Обычно аналитики требований начинали собеседование или семинар с вопроса к пользователям: «Чего вы хотите?» Это самый бесполезный вопрос в процессе сбора требований. Второй по бессмысленности вопрос: «Какие у вас требования?» Люди не вполне понимают, как отвечать на столь туманные вопросы, и часто дают хоть и важную, но неструктурированную информацию. В этом проекте мы не хотим попасть в ту же ловушку, поэтому нам нужен лучший способ выследить этого скрытного и осторожного зверя – требования.

Методика вариантов использования

Некоторые сотрудники нашей ИТ-группы постоянно посещают технические конференции по разработке ПО. Затем на еженедельном собрании группы они в общем виде рассказывают об услышанном, и мы все вместе думаем, каким образом можно применить новые методы в наших проектах. Я вспоминаю о том, что один мой коллега недавно смотрел презентацию метода сбора требований под названием *варианты использования* (use cases). Звучит многообещающе для нас, и я беру у него копию презентации.

Просматривая его заметки по этой методике, я начинаю понимать, что она, кажется, подходит и для системы учета химреагентов. У нас под рукой не так много информации об этом, и никто из нас не пользовался методикой прежде, но общая концепция осмысленна и понятна. Основная идея этой методики заключается в том, что акцент должен делаться не на функциях и возможностях самой системы, а на том, каких целей сможет достичь *пользователь* при помощи этой системы. То есть в центре нашего подхода будет не сам продукт, а его применение. Поэтому вместо «Чего вы хотите?» или даже «Чего вы хотите от системы?» мы будем спрашивать «Что вам нужно будет сделать при помощи этой системы?». Маленькое изменение в постановке вопроса – и значительное изменение общей концепции. Каждая из этих «вещей, которые пользователь хочет делать при помощи системы», и есть потенциальный вариант использования.

Поскольку прежде этим методом мы не пользовались, то не знаем, как он работает, и будет иметь место «кривая приобретения навыка». Не знакомы с этим методом и наши сторонники продукта. Потребуется объяснить им их роль при новом подходе и рассказать, почему мы предпочли этот путь более традиционным обсуждениям требований.

Семинары

Вместо того чтобы проводить много отдельных собеседований с представителями пользователей, мы решаем провести ряд семинаров по упрощенному сбору требований. Каждый аналитик будет периодически встречаться со своими представителями пользователей, чтобы узнать у них о целях, которые они планируют достичь при помощи системы учета. На каждом таком семинаре аналитик выступит в качестве координатора обсуждения и секретаря, ведущего запись (хотя вскоре мы поняли, что иногда полезно разделять эти две обязанности). Такое общее собрание пользователей часто позволяет сэкономить время на разрешение конфликтов, возникающих, когда собеседования с сотрудниками проводятся в индивидуальном порядке и по очереди. Также при этом достигается синергетический эффект, помогающий участникам генерировать новые идеи и понимать общую концепцию будущего продукта.

Мы решаем проводить семинары отдельно по классам пользователей. Поскольку мы разделили всех пользователей на несколько классов с разными требованиями, совместный семинар не имеет смысла.

Я веду семинары для представителей химиков (сторонник продукта Сара и пять участников группы поддержки). Чтобы согласовать общую стратегию взаимодействия группы, мы устанавливаем некоторые основные правила для всех семинаров. Одно из главных правил: семинар начинается и заканчивается в назначенное время, опаздывающих не ждем. Второе правило: те, кто пришел на семинар, составляют кворум для принятия решения. Если кто-то не может прийти, ему предлагается сообщить свое мнение заранее либо направить делегата. Также у всех есть возможность просмотреть результаты каждого семинара. Но присутствующие имеют право принимать важные решения, таким образом, мы можем двигаться вперед с необходимой скоростью. Согласование этих и других основных правил – важный элемент создания эффективного командного сотрудничества.

До первого семинара я предлагаю химикам подумать о том, «что именно нужно будет делать химикам при помощи системы учета». Каждая из определенных таким образом задач станет потенциальным вариантом использования, которые мы затем подробно рассмотрим на семинарах. Есть большие задачи (заказать химреагент), есть задачи меньшего масштаба (просмотреть размещенный заказ). Я назначаю дату нашего первого семинара и планирую начать с вариантов использования системы, которые кажутся наиболее важными.

Не вполне понимая, что делать дальше, снова заглядываю в упомянутую презентацию. Докладчик предлагает записывать ключевые эле-

менты варианта использования на плакатах и стикерах. Это такие элементы, как:

- Название варианта использования.
- Действующее лицо данного варианта использования (в первом приближении, *действующее лицо* – это тип пользователя).
- Краткое описание варианта использования и его приоритетность.
- Необходимые предварительные условия для осуществления системой этого варианта использования.
- Условия на выходе – что получается при успешной реализации варианта использования.
- Обычный порядок действий, который описывает наиболее типичный набор взаимодействий пользователя и системы, приводящих к успешному результату.
- Альтернативный порядок действий, который описывает другие способы действий пользователя в данном варианте использования (опции ветвления и др.).
- Исключения, то есть возможные случаи, в которых данный вариант использования может не быть успешно реализован.

Мне нравится, что этот подход начинает оправдывать себя на семинарах. Еще больше мне нравится видеть, что мои химики поняли и приняли подход с вариантами использования. Он удобен для многих пользователей, поскольку напрямую связан с их работой и тем, как они планируют применять систему в своей работе. По ходу дела я учусь вести семинары более эффективно, ограничивать время их проведения двумя с половиной часами и задавать четкие временные рамки для обсуждения каждого варианта использования, чтобы мы не увязали в деталях. Я также начинаю понимать, что необходимо поддерживать обсуждение на нужном уровне абстракции. Мы стараемся не зарываться в излишние подробности, например не начинать заранее обсуждать особенности пользовательского интерфейса.

Чувствую себя отважным исследователем и первооткрывателем этого метода. До нас этот метод не применялся, и я стараюсь как могу, прокладывая новый путь. Когда мои химики входят в ритм работы и у нас начинает получаться, другие аналитики приходят на наш семинар и решают, каким образом они смогут применить этот метод в своих группах. Джанет и Девон проводят свои семинары подобным же образом и также достигают успехов со своими пользователями. Три аналитика работают отдельно, но мы делимся опытом и открытиями, и каждый учится работать более эффективно. В любой команде разработчиков ПО работа может стать более продуктивной, если они будут учиться у своих коллег.

Одних вариантов использования недостаточно

Анализируя результаты одного из семинаров по вариантам использования, я понимаю, что у нас хорошо получилось описать процесс взаимодействия пользователей с системой учета для достижения различных целей. То есть такой метод хорош для описания *требований пользователя*. Однако он не дает разработчику всю информацию, необходимую для реализации соответствующих возможностей в системе. Кроме того, при таком способе информация разбита на отдельные элементы, и передавать ее в таком виде разработчику не очень удобно. Мы приходим к выводу, что один метод вариантов использования не даст нам полностью собрать необходимые требования.

Чтобы выйти на следующий уровень, нужно вывести соответствующие *функциональные требования* по каждому варианту использования. Как бывший разработчик я понимаю, что разработчики не могут реализовать варианты использования как таковые. Они реализуют определенные детали работы системы – функциональные требования, которые вместе дают пользователю возможность работать с определенным вариантом использования. Поэтому после каждого семинара я анализирую собранную информацию и начинаю составлять спецификацию требований по программному обеспечению (СТПО). По каждому варианту использования я прописываю набор функциональных требований, который, в случае его реализации, позволит пользователю достичь цели, описанной в данном варианте использования.

Некоторые из этих функций напрямую следуют из описания, например те, что относятся к диалогу между пользователем и системой. Однако из описания варианта использования можно вывести и дополнительные функции, не столь очевидные. Например, в предварительных условиях по вариантам использования никак не описано, что должна делать система в случае, если предварительные условия *не* выполняются. Некоторые действия системы невидимы для пользователя, поэтому и вариант использования не может их описать. Данные функциональные требования более полно описывают ожидаемое поведение системы, чем то, что получили бы разработчики в случае, если бы мы просто передали им пакет вариантов использования и велели: «Когда будет готово, сообщите».

Не только функциональность

Люди лучше всего учатся на ошибках и отрицательном опыте. Вспоминается предыдущий проект, где мы замечательно прописали функциональные возможности, но недостаточно хорошо исследовали ожидания пользователей в области качества. В результате новая система встретила значительное сопротивление со стороны пользователей, поскольку

соотношение эффективности, удобства в применении и других характеристик было далеко не идеальным. Чтобы не повторить тот печальный опыт, в рамках определения требований к системе учета мы обсуждаем также требования к *качественным показателям* системы.

Качественный показатель описывает не *что* делает система, – это относится к функциональным возможностям, – а *насколько хорошо* она это делает. К качественным показателям относятся: удобство пользования, портируемость, надежность в эксплуатации, простота установки, готовность к работе, качество работы, эффективность, ошибкоустойчивость и защищенность. У пользователей есть подспудные ожидания по определенным качественным характеристикам. К сожалению, зачастую они не сообщают сразу же об этих своих ожиданиях. Пользователям сложно сформулировать свои ожидания в отношении качества продукта таким образом, чтобы помочь проектировщикам и разработчикам.

Недостаточно прямо спросить у пользователя: «Каковы ваши требования к ошибкоустойчивости системы?» – этот вопрос не даст нужную информацию. И мы пробуем другой подход. Мы думаем о том, что может значить термин «ошибкоустойчивость» для наших пользователей, а затем пишем несколько вопросов, направляющих ход мысли пользователя к определенным аспектам ошибкоустойчивости. Например: «Как по-вашему, что должна делать система, если в процессе размещения нового заказа на химреагент происходит сбой сети?» И так же поступаем в отношении других показателей качества, которые могут стать существенными для успешной работы системы. Мы, аналитики, просим пользователей ответить на эти вопросы и оценить относительную важность каждого показателя с их точки зрения. Таким образом, мы получаем картину, позволяющую понять, какие характеристики системы критически необходимы, а какие имеют меньшее значение. Эти требования к показателям качества закладываются в нашу СТПО, и они не менее важны, чем описание функциональных возможностей.

Правила игры

В ходе одного из семинаров по сбору требований химики обсуждали вариант использования под названием «просмотр размещенного заказа». Кто-то из группы поддержки сказал: «Я не хочу видеть заказы, размещенные Сарой, а Сара не должна видеть мои заказы». Химики согласились с тем, что это хорошая политика. И мы записали это как *деловое правило (business rule)*. Такие правила могут составляться на основе разных источников:

- Политики компании, подразделения или проекта, например политики доступа к информационным системам.

- Законы и положения, например по отчетности по обращению с химреагентами и их уничтожению.
- Промышленные стандарты, такие как форматы файлов для приема и отправки химических структур.
- Алгоритмы расчетов, например для определения скидки на приобретение крупных партий у одного поставщика.

Деловые правила сами по себе не являются требованиями к ПО. Однако такое правило может определить функциональную возможность, необходимую для того, чтобы система работала в соответствии с определенной политикой. Правило о том, кто и какие заказы может просматривать, определяет необходимость функциональной возможности идентифицировать каждого пользователя, прежде чем разрешать ему просмотреть заказ. В ходе семинаров по сбору требований каждый класс пользователей определяет (или разрабатывает) целый ряд деловых правил. Эти правила помогают обрисовать рамки, в которых затем уже смогут работать разработчики ПО.

Проверка до начала программирования? Вы в своем уме?

Требования описывают то, что мы должны получить, когда работа будет сделана. Но когда я работал над плакатом после одного из семинаров, меня озарило. В случае метода вариантов использования предполагается, что процесс работы естественным образом приходит к тому, что нам необходимо определить испытания, позволяющие понять, разработали ли мы именно то, что планировалось. Я никогда прежде не слышал об этой связи между вариантами использования и проверкой, но это была здравая мысль.

И я начал по описаниям вариантов использования прописывать проверки, позволяющие понять даже не то, как реализовать вариант использования, а то, каким образом можно проверить, правильно ли была выполнена эта реализация. Испытание на концептуальном уровне, не вдаваясь в подробности конкретной реализации и пользовательского интерфейса. При написании этих проверок я обнаружил некоторые ошибки в своем варианте использования. Это было еще одно открытие. Возможно, я смогу применить эти концептуальные проверки, разработанные на ранней стадии формирования требований, для обнаружения ошибок задолго то начала какого-либо проектирования или программирования!

Я задумался над этим. Как уже говорилось, я выводил функциональные требования из описания вариантов использования. В принципе,

если разработчик реализует эту функциональность, пользователь сможет применять вариант использования. Теперь на основе того же варианта использования, но по другой логике я вырабатываю тесты. И сравниваю эти тесты с моими функциональными требованиями, пытаясь определить:

- Есть ли тест для каждого из моих функциональных требований?
- Сможет ли каждый из моих тестов быть «выполнен» посредством определенного набора функциональных требований?

Проводя такой анализ, я обнаруживаю недостающие, неверные или излишние требования. Некоторые из них указывают на проблемы с вариантом использования, другие – на неправильное разбиение мною этого варианта на функциональные элементы. Кроме того, я нахожу недостающие, неверные или излишние тесты. Устранив все обнаруженные проблемы, я прихожу к уверенности в том, что мои требования верны.

Я понимаю также, что было бы даже лучше, чтобы функциональные требования и тесты на основе варианта использования разрабатывали разные люди с разным мышлением. Разночтения между результатами, полученными этими людьми, могли бы указать на двусмысленности и упущения в варианте использования, которые могут повлечь разное толкование. Теперь я знаю, что к тестированию программного приложения можно приступать сразу после того, как написано первое техническое требование к нему. Такой подход несомненно лучше тестирования после разработки, когда на устранение ошибок в техническом задании потребуется гораздо больше времени и денег.

Я прошу моего сторонника продукта Сару проверить тесты, которые я написал по вариантам использования для класса химиков. Она предлагает дополнительные поправки и улучшения. На завершающем семинаре все шесть представителей химиков проходят по всем тестам, чтобы убедиться в общем для всех понимании того, как будет работать система учета. У нас могут быть разные представления о том, что выводится на экран, и этот вопрос мы сможем проработать позже, на этапе проектирования. А в данное время мы уверены, что все химики одинаково понимают, чего они ожидают от системы учета химреагентов.

Акт III: проверить друг друга

Мы проводим много технических экспертиз, чтобы выявить максимум ошибок в требованиях.

Аналитику недостаточно просто документировать те требования, которые он слышит на собеседованиях и семинарах, чтобы быть уверенным в их корректности. Есть столько возможностей для неправильного по-

нимания или толкования, что проверка и подтверждение правильности – необходимый этап процесса разработки технического задания. *Экспертная оценка (peer review)* представляет собой один из наиболее мощных механизмов обнаружения ошибок в требованиях. В ходе такой оценки кто-то, но не сам автор документа, проверяет выполненную работу на наличие недостатков и возможность улучшения.

Экспертная оценка является своего рода статическим испытанием, возможностью отфильтровать погрешности в техническом задании прежде, чем будет написана первая строчка кода. Такие проверки дают пользователям возможность подтвердить, что предоставленные ими сведения были верно записаны и истолкованы. Они также позволяют определить двусмысленность некоторых требований, после чего заинтересованные стороны могут разобраться и прийти к общему пониманию того, что подразумевается этим требованием. Если бы в проекте разработки ПО мне был доступен только один вид проверки качества, я бы провел формальную экспертную оценку (также называемую *технической экспертизой*) всей информации, связанной с требованиями. Понимая всю мощь экспертных оценок, мы проводим их в рамках разработки требований в двух вариантах, неформальном и формальном.

Неформальный вариант

После каждого семинара по сбору требований аналитик передает все созданные по его результатам документы участникам семинара. Такие документы включают варианты использования, фрагменты разрабатываемой СТПО, графические аналитические модели, описание данных и другие сведения, полученные в ходе семинара. Участники самостоятельно неофициально проверяют эти документы на наличие ошибок, упущений, неправильных толкований и других проблем.

И, поверьте, при этом обнаруживается очень много ошибок! Иногда аналитик неправильно понимает что-то или по-своему истолковывает какое-то требование, и пользователи замечают это и исправляют. Или пользователь замечает, что мы не учли альтернативный порядок действий по данному варианту использования. Или он видит, что к ошибочно прописанному условию нужно подойти иначе, чем мы думали первоначально. Иногда в процессе проверки приходит идея какой-то дополнительной функциональной возможности, полезной для пользователей. Эти самостоятельные, пошаговые, неофициальные и недорогие оценки значительно повышают качество нашего растущего объема информации по требованиям. Типы обнаруживаемых нами проблем дают нам также понимание того, как можно улучшить в будущем наш процесс выявления требований и разработки технического задания. Кроме того, эти оценки всерьез убеждают Пола и другие заинтересован-

ные стороны, что мы делаем ощутимые успехи в понимании и регистрации их требований к системе учета химреагентов.

Аналитики из нашей команды пробовали по-разному организовывать экспертную оценку. Я провожу раз в неделю семинары со своими химиками. В течение двух дней после каждого семинара я выдаю результаты своей работы стороннику продукта и группе поддержки. Проверка этой документации позволяет тем, кто не принимал участие в семинаре, догнать остальных. Мои пользователи замечательно справляются с обнаружением ошибок и делятся своими находками со мной в начале каждого следующего семинара.

Самый молодой из аналитиков, Девон, подходит к вопросу по-другому. Он проводит ежедневные семинары со своими пользователями, поэтому ему приходится очень быстро фиксировать результаты каждого семинара и передавать их участникам для проверки до начала встречи на следующий день. Это заставляет всех участников очень спешить. И, что еще важнее, Девон сообщает, что его пользователи находят не очень много проблемных вопросов. К сожалению, проблемы там есть, просто проверяющие обнаруживают не все из них.

Идея здесь в том, что когда вы просматриваете документ вскоре после того как напряженно думали над ним (либо в процессе создания как автор, либо в качестве участника семинара), вы на самом деле не проверяете документ, а пытаетесь в уме восстановить то, что было. Скорее всего, вы обнаружите меньше ошибок, чем если бы у вас было немного времени на то, чтобы эта информация «осела», прежде чем вы вернетесь к ней. И наоборот, если вы возвращаетесь к документу через один-два дня, когда он уже стерся из краткосрочной памяти, вы можете посмотреть на него более свежим взглядом и под другим углом. С большей долей вероятности вы сможете обнаружить несоответствия, упущения и другие проблемы. Возможно также, что эти два дня вы подспудно думали над этим вопросом и у вас появились идеи о том, как улучшить первоначальный вариант. Осознав это, я всегда стараюсь отложить созданные мною документы хотя бы на 24 часа, прежде чем приступить к самостоятельной проверке. Это позволяет мне более внимательно проверить их, чем если бы я делал это сразу после написания.

Формальный вариант

Тот тип неформальных проверок, который я описал в предыдущем разделе, можно назвать *вычиткой*. Проведение нами вычитки позволяет многое улучшить, но не может заменить более строгих и тщательных командных экспертных проверок, называемых *технической экспертизой*. В ходе экспертизы несколько участников проверяют результаты работы самостоятельно, а затем обнародуют свои замечания и вопросы на

общем собрании. Обсуждения в ходе экспертиз часто приводят к выявлению проблем, не замеченных проверяющими в ходе самостоятельной работы.

Экспертиза технического задания дает еще одно преимущество – возможность выявить неясности и двусмысленности. Двусмысленное требование – такое, которое может быть истолковано разными читателями по-разному (а иногда и одним читателем). Допустим, каждый проверяющий в ходе самостоятельной вычитки видел такое двусмысленное требование. Само по себе оно имеет для него смысл, но смысл этот отличается для разных проверяющих. Каждый из них скажет, что все в порядке, и двусмысленность так и останется незамеченной. Однако в ходе технической экспертизы один из членов команды, называемый *читателем*, описывает своими словами понимание каждого требования. Другие проверяющие могут сравнить эти объяснения с собственным пониманием требований. Иногда таким образом выявляется разница в толковании – двусмысленность, чреватая серьезными проблемами, если не устранить ее на раннем этапе.

С учетом столь мощных преимуществ мы решаем провести экспертизу составленной нами СТПО. Джанет берется объединить технические задания и сопутствующие материалы, разработанные тремя аналитиками. В итоге мы получаем 50-страничный документ СТПО, к которому прилагается еще 50 страниц вспомогательной документации, включая модели анализа, определения и прочее.

К технической экспертизе мы привлекаем наших сторонников продукта, всех трех аналитиков и еще одно заинтересованное в проекте лицо. Восемь человек – это внушительная экспертная команда. Я проводил экспертизы в течение нескольких лет, поэтому выступаю в роли председателя, помогая вести дискуссию в нужном русле. Девон выступает секретарем группы, записывая все, что выясняется в ходе обсуждения. Мы понимаем, что физически не сможем за одну встречу перелопатить такой объем материала, если хотим тщательно выявить проблемы. Поэтому мы назначаем три встречи экспертной группы: в понедельник, среду и пятницу одной недели, каждая на два с половиной часа (и это уже достаточно быстрый темп технической экспертизы). Устраивать марафонские проверки бессмысленно, потому что после пары часов усталые головы уже не способны к новым озарениям.

Наша экспертиза проходит успешно, по крайней мере, если измерять успешность количеством найденных ошибок. Каждая исправленная сейчас ошибка означает, что нам не придется устранять ее позже, когда это будет гораздо дороже и сложнее. Девон несколько раз бегает за дополнительными бланками для записи найденных проблем. Обнаружить столько проблем после упорного труда над требованиями не

очень-то приятно. Но мы понимаем также, что обнаружить все их на этом этапе – удача.

Результат

В ходе экспертизы мы выявляем целых 223 неточности и вопроса. Большинство из них мелкие, но некоторые могли бы серьезно повлиять на проект, не будучи обнаруженными на раннем этапе. Выявив столько ошибок, мы очень рады, что затеяли экспертизу. Очевидно, что затраты на проведение экспертизы, хоть и значительные, гораздо меньше потенциальных потерь, к которым эти ошибки могли привести, будучи переданными в рамках технического задания разработчикам и тестерам.

После экспертизы мы исправляем эти многочисленные ошибки. Пол и другие сторонники продукта подтверждают, что исправленные документы в точности отражают их требования к системе учета химреагентов на текущий момент. Следующим шагом мы создаем *целевой план (baseline)* по нашей СТПО. Наличие целевого плана не означает, что технические условия зафиксированы и не могут быть изменены в будущем. Фактически, ТУ *обязательно* изменятся, по многим причинам. Заказчики вспомнят то, о чем не подумали, у аналитиков возникнут гениальные идеи по функциональности, в ходе проектирования и программирования могут обнаружиться другие ошибки в требованиях, да и сами условия ведения бизнеса могут измениться за период разработки. Но наш целевой план служит согласованным основанием для последующей работы по проекту. И хотя мы осознаем, что техническое задание еще не идеально, его создание является значительной вехой и дает всем участникам ощущение того, что мы достигли своей цели по этому серьезному проекту. Разработанное техническое задание «достаточно хорошо» для того, чтобы приступить к следующему этапу разработки системы учета химреагентов.

Эпилог: не пора ли подкрепиться?

В ходе работы над сбором требований мы замечаем, что лед в отношении Пола к нам, с которым мы столкнулись на начальном этапе, начинает таять. Он видит, что мы продуктивно работаем и достигаем успехов в понимании его требований и требований других пользователей системы учета. Пол отмечает, что применяемые нами новые методики позволяют лучше определить необходимые требования, чем подход, принятый нашими предшественниками. В результате Пол уверен, что окончательный вариант технических условий, воплощенный должным образом, позволит ему выполнить предписания государственных органов, а также будет ценным подспорьем для многочисленных пользователей.

Мы также отмечаем явные признаки изменения самой культуры взаимодействия в результате подхода к составлению требований по системе учета, примененного нашей командой. Вскоре после создания целевого плана по техническому заданию Пол дал роскошный обед для команды аналитиков, сторонников продукта и других ключевых участников. Столы ломились от яств, все отлично провели время и никто не ушел голодным. Пол значительно повеселел по сравнению с предыдущими четырьмя месяцами.

Кода: что было потом

Став в 1998 году независимым консультантом, я большей частью работаю с клиентами только какое-то время, на определенном этапе. Провожу обучение, консультации или оказываю помощь по возникшей конкретной проблеме. И часто не вижу, как сказывается моя работа на дальнейшем развитии проекта. То же было и с проектом системы учета – поначалу.

После успешного составления технического задания я занялся другим делом и вскоре перешел в совсем другое подразделение Contoso Pharmaceuticals. Два года спустя я ушел из Contoso и основал собственную консалтинговую и тренинговую компанию в области ПО. На обучающих курсах я пользовался некоторыми примерами по системе учета химреагентов, чтобы проиллюстрировать методику и открытия в области составления технических заданий.

В 1999 году я читал курс лекций по сбору и разработке требований на предприятии заказчика. Когда я начал рассказывать о работе над системой учета, один из слушателей сказал: «Я узнаю этот проект!» Оказалось, что он занимается маркетингом, и прошлым его работодателем была компания, которой Contoso поручила разработку некоторых элементов системы учета. Заинтригованный таким совпадением, я спросил у него, как шла реализация проекта и что думали разработчики той его компании о техническом задании на эту систему. Большим облегчением для меня было узнать, что проект был успешным и немаловажную роль в этом сыграло грамотное техническое задание.

Несмотря на неудачное начало ее истории, система учета химреагентов стала показательным примером. Мы научились выборочно и продуманно применять новые методы в разработке технических условий. Нашли способы задействовать в процессе разных представителей заказчика. Нам удалось достичь положительных изменений в культуре работы, поскольку Пол и другие ключевые заказчики поняли, что на самом деле возможно и даже необходимо тесно сотрудничать с ИТ-специалистами, чтобы гарантировать разработку требуемого продукта. Самое прият-

ное, что эта система все еще используется в Contoso Pharmaceuticals, через 13 лет после описанной здесь истории.

Полезные ссылки

1. Ambler, Scott, 1995, «Reduce Development Costs with Use-Case Scenario Testing», *Software Development* 3(7):53–61.
2. Collard, Ross, 1999, «Test Design», *Software Testing and Quality Engineering* 1(4):30–37.
3. Gottesdiener, Ellen, 2002, «Requirements by Collaboration: Workshops for Defining Needs», Boston: Addison-Wesley.
4. Kulak, Daryl, and Eamonn Guiney, 2004, «Use Cases: Requirements in Context», Second Edition, Boston: Addison-Wesley.
5. Wiegers, Karl E., 1996, «Creating a Software Engineering Culture», New York: Dorset House Publishing.
6. Wiegers, Karl E., 2002, «Peer Reviews in Software: A Practical Guide», Boston: Addison-Wesley.[1]
7. Wiegers, Karl E., 2003, «Software Requirements», Second Edition, Redmond, WA: Microsoft Press.
8. Wiegers, Karl E., 2006, «More About Software Requirements: Thorny Issues and Practical Advice», Redmond, WA: Microsoft Press.

Благодарности

Автор выражает признательность Джиму Броссо, Барб Карденуто, Крису Фальбушу, Кэти Гетц, Андре Гусу, Шеннон Джексон, Лори Ширмер и Мо Станкай за ценные замечания и комментарии.

[1] Вигерс К. «Разработка требований к программному обеспечению». – Пер. с англ. – М.: Русская редакция, 2004.

20

Разработка в Google

Интервью с Алексом Мартелли

Похоже, все наши знакомые хотели бы работать в Google. Мы слышали о многом: что разработчики могут там часть времени работать над собственными проектами, что в штате компании есть массажисты, которые поддерживают всех в добром здравии. Google очень серьезно подходит к разработке программного обеспечения, и Алекс Мартелли, изначально принятый в компанию на должность главного технического руководителя, занимался важной стороной разработки ПО. Мы попросили его рассказать о тимбилдинге в Google.

Эндрю: **Как вы приобрели такие знания о работе ПО?**

Алекс: Я начинал как инженер-электрик: проектировал чипы для компьютеров в Texas Instruments, а затем в IBM Research. Постепенно от оборудования, от чипов, я перешел к системе, к микропрограммированию, потому что исследовательский отдел в то время просто строил прототипы – чтобы их могла использовать наша целевая аудитория, то есть ученые и так далее, они должны были поддерживать, например, FORTRAN или APL. Микрокод им был совсем не нужен. И вот, несколько лет продвигаясь в эту сторону, я вдруг понял, что, собственно, уже не проектирую оборудование. Я перестал ориентироваться в том, что происходит в этой сфере – в чипах и в технологиях, и пришлось смириться с тем, что я стал программистом.

Вскоре после этого я поступил в стартап-компанию, где, по идее, все должны были заниматься всем. Поскольку я был самым опытным разработчиком в группе, пришлось стать менеджером, то есть я должен был организовать команду и наладить ее работу. Сначала ничего не получалось. К счастью, не успев угробить компанию, я нашел того, кто хоть что-то понимал в менеджменте, а сам вернулся к программированию. Позже, учтя свои ошибки, я снова попробовал руководить. Постепенно пришел опыт, и я научился сочетать чисто технические навыки с организационными, лидерскими и менеджерскими.

В 2004 году я прошел собеседование в Google и под конец года получил от них приглашение на заманчивую должность главного технического руководителя. То есть заниматься надо было тем же, что я делал последние несколько лет своей жизни, – как техническими аспектами, так и руководством и лидерством. Слово «главный» означает, что я не отвечал ни за одну из команд разработчиков Google – у каждой из них был собственный технический руководитель. Моей же задачей было координировать и интегрировать несколько команд в определенной области.

Эндрю: **Давайте немного поговорим о вашем первом опыте руководства. Как это было?**

Алекс: Думаю, в моих неудачах в то время в основном виновата коренная черта моего характера – твердая вера в человека. Слово *вера* я употребил здесь не случайно: тут можно только верить. Кто видел жизнь, не станет утверждать, что основа человеческой натуры – добро. Слишком много свидетельств обратному. Думаю, я просто верю в это.

Например, у нас был парень, которого взяли, потому что в предыдущем проекте он работал с одним из основателей. Но тот основатель не был программистом, и никто не понял, что на работу приняли человека, который был просто не готов к тому усиленному давлению, какое

испытываешь, работая в стартапе. Возможно, он был хорош в крупной компании, где имел ограниченные обязанности и достойно выполнял их, не создавая проблем. Но чтобы стартап выжил, нужно работать на порядок интенсивнее, увлеченнее, обладать большими навыками. И самым ценным, что он вынес из своего предыдущего опыта работы в крупных компаниях, было умение убедительно оправдываться.

Эндрю: **Забавно. Но, может, вы малость преувеличиваете?**

Алекс: Нет, он не жаловался на то, что его программы съела кошка, но мы продолжали попадаться на его удочку раз за разом. И в итоге получился... не то чтобы бунт, но все же остальные члены команды не были дураками. Да и я, собственно, тоже был не дурак – просто доверчивый оптимист. А они понимали, что из-за этого парня вся работа идет насмарку. Его продуктивность была отрицательной. Он мог работать интенсивно. Он не увиливал от работы, но когда он три дня работал в нашем режиме, потом кто-нибудь семь дней исправлял все его ошибки. Я давал ему второй, третий, четвертый шанс. Сейчас-то я понимаю, что это звучит глупо. Но поверьте, что из-за всего объема дел, которые нужно было переделать за то время, только сейчас до меня доходит, что я делал не так.

В итоге мне пришлось вызвать этого парня и сделать то, что для руководителя труднее всего, но что является самым важным для хорошей работы команды. Мне пришлось сказать: «Да, ты был с нами с самого первого дня, но дальше так нельзя». Впервые в жизни я был вынужден кого-то уволить, ну или предложить ему поискать другие возможности, и совершенно не ожидал, насколько это болезненно. Но, в конце концов, это было абсолютно необходимо, чтобы команда выжила. Одна паршивая овца может испортить все стадо. Если руководство слишком доброжелательно, если команда сформирована на основе знакомств и приятельских отношений, то произнести эти недобрые, но справедливые и необходимые слова тому, кого ты считаешь личным другом, еще сложнее.

То, что он не умел хорошо программировать, не означает, что он плохой человек. Более того, уверен, что его оправдания казались мне настолько убедительными не только из-за моего неисправимого оптимизма, но и из-за его искренней уверенности в том, что это и есть истинные причины его неудач. В итоге я поступил правильно. Уверен, что можно научиться не испортить при этом дружбу, но я, к сожалению, не умею. Тут как с личными отношениями – когда они не клеятся, а вы продолжаете предоставлять шанс за шансом. Это может разбить вам сердце, потому что если этот человек часть вашей души, обижать его не хочется. Но если ничего не выходит, то что остается? Пожалуй, за всю жизнь это был самый суровый урок, который мне пришлось усвоить.

Дженни: **То есть, по-вашему, проблемы в работе команды заключались в этом конкретном человеке и его вечных оправданиях?**

Алекс: Именно так. Все вкладывали много сил в успех предприятия, ведь это был стартап. Зарплаты были на уровне карманных денег, а большой доход должен был прийти с победой фирмы на рынке. Поэтому люди были профессионально заинтересованы в успехе фирмы, от этого зависела вся их жизнь. И оставлять в команде человека, который снижал шансы на успех, было просто бессмысленно. Попробую объяснить: разве работа руководителя не в том, чтобы это понимать? Именно в этом. А я плохо делал свою работу, то есть часть своей работы. Мне, конечно, было чем заняться и помимо этого – руководство, технические вопросы, – но в этом отношении я определенно недоработал, и это было важно.

Остальная часть команды была идеальна. Когда я снова стал одним из многих и нашел себе на замену хорошего парня, которому можно было вручить бразды правления, все пошло на лад. Во-первых, теперь 100% своего времени и энергии я мог тратить на разработку программ, а во-вторых, этот парень был, разумеется, намного лучше и опытнее как руководитель, чем я в то время.

Эндрю: **Что бы вы посоветовали члену команды программистов, заметившему, что у его руководителя подобная проблема? Могли ли члены вашей команды помочь вам осознать, насколько плоха ситуация, чтобы вы скорее ее исправили?**

Алекс: Тут масса вариантов: поверьте, я много размышлял над этим. Уверен, что не вводил себя в заблуждение сознательно. Обычно очень хочется думать, что ничего не происходит, что все само рассосется, как если бы кто-то не хотел идти к зубному: человек подозревает, что у него с зубами что-то не так, но боится боли и никуда не идет. Это неправильно, но это в порядке вещей. Мы не любим сталкиваться с суровой реальностью – это может привести к болезненным действиям и последствиям.

Итак, допустим, кто-то думает: «Нас в команде шестеро, но будет гораздо лучше, если нас останется пятеро, потому что от этого парня один вред вместо пользы, а начальник этого совсем, совсем не видит!» Особенно важно это в небольшой фирме или в стартапе, когда успех команды критичен для каждого. В том числе и для руководителя.

Как сказать руководителю? Это столь же непросто, как самому руководителю сообщить плохому работнику об увольнении, особенно наедине. Если вы понимаете, что ваш товарищ по команде не справляется с нагрузкой, значит это мог понять и кто-то еще. Когда взгляд на ситуацию руководителю излагают двое или трое, это может оказаться более эф-

фективным вариантом. Менеджер, скорее всего, поверит, так что если вы можете позвать двух-трех товарищей по работе, то так и поступите.

Кстати, здесь лучше говорить лицом к лицу. Просто не представляю, как можно уволить сотрудника по электронной почте, телефону или онлайн-мессенджеру, потому что сообщая плохую новость, нужно оставить человеку возможность выплеснуть эмоции с помощью жестов и непосредственного общения. В то же время крайне важна эмоциональная поддержка: «То, что вы плохо делаете свою работу, вовсе не означает, что вы плохой человек. Просто здесь вы не преуспели и должны заняться чем-то другим». Все это лучше сообщить в личной беседе, как и большинство плохих новостей.

Дженни: **Как вы считаете, насколько успех или неудача той команды зависели от вас как от руководителя, а насколько — от самой командной динамики?**

Алекс: Скорее все же успех зависел от руководителя, потому что успех руководителя – это и есть успех команды. Просто руководителем должен был быть не я. И действительно, как только я стал ведущим разработчиком, передав бразды правления другому, – все пошло как по маслу. Дело было около 20 лет назад, но не думаю, что человеческая природа с тех пор сильно изменилась. Технология – да. Человек – вряд ли.

Да, появились технологические и методологические аспекты, которые очень помогают в работе: только представьте, что 20 лет назад у нас уже была бы четко структурированная Agile-организация команды с ежедневными собраниями. Огромный плюс Agile-методики и вместе с тем большой риск (потому-то многие и отказываются брать ее на вооружение) заключается в том, что истина в этом случае выясняется куда быстрее, чем при традиционных каналах общения.

Если утром каждый вкратце рассказывает, чего достиг вчера, что собирается сделать сегодня и какие у него проблемы, с которыми ему может помочь кто-то другой, то долго плохую работу не скроешь. Все становится вполне очевидным. Это всего один пример того, как Agile-методы обеспечивают лучшую взаимосвязь коллектива, не позволяя – приятно это или не очень – скрывать информацию от других. Классический пример – четкая картина хода работ, которую дают Agile-оценки, например диаграммы burn-down. Все очевидно: «Вот, ребята, это наша скорость. Вот доступный объем работ. Если сохранить все предполагаемые функции, то опоздаем на два месяца». Это типично для многих программных проектов, но не все готовы с этим смириться. Можно работать сверхурочно и даже сгореть на работе, но в итоге это все равно лишь отсрочка. Если выяснить истину слишком поздно, то неудача неизбежна.

Наглядность Agile-методов позволяет выяснить неприятное положение дел намного быстрее. Чем раньше взглянуть в лицо действительности, тем легче исправить ошибки. Возможно, придется пожертвовать рядом функций и передвинуть на пару недель срок сдачи, но только если неполадки выяснились достаточно рано. Но если все выяснилось уже после дедлайна, то слишком поздно принимать такие относительно безболезненные меры.

Эндрю: **Как вы считаете, есть ли у Agile-подхода недостатки?**

Алекс: Недостаток в том, что ничего нельзя скрыть, и для многих разработчиков и менеджеров это звучит угрожающе. Бывает, проект на грани провала, но в больших корпорациях обычно получается так, что за две недели до предполагаемого срока сдачи случается большая реорганизация, проект куда-то исчезает, и в вашем личном деле не остается пятна. В каком-то смысле это подобно выигрышу в лотерею, но на это продолжают надеяться, чтобы избежать вреда своей карьере. Не думаю, что с точки зрения команды, корпорации или заказчика это большой минус, но для менеджера и разработчиков команды ситуация тревожная: ничего не скрыть. Для всех, кто причастен к проекту, положение дел абсолютно прозрачно.

Еще один момент, уже упомянутый в связи с плохой новостью, – по моему скромному опыту, Agile-методика вряд ли применима в распределенной команде. Я знаю тех, кто утверждает обратное, и восхищаюсь ими, но если команда рассредоточена, то все преимущества взаимосвязи утрачиваются. Могу сказать только: при распределенной команде не поможет никакая магия. Подозреваю, что Scrum-разработка просто не годится для этого.

Эндрю: **Интересно, что вы подчеркнули два очень важных для команды момента: прозрачность, то есть невозможность что-то скрыть, и способность поддерживать постоянную взаимосвязь.**

Дженни: **Думаю, все методики разработки, не только Agile, во многом основываются на прозрачности, на возможности убедиться, что все понимают, кто за что отвечает и как продвигаются дела. Вы сказали, что методы оценки вроде диаграмм burn-down помогают наглядно представить информацию. Пожалуйста, поясните, почему именно эти методы предпочтительнее, скажем, освоенных объемов или других подходов, которые десятилетиями использовались как в программных, так и в других проектах.**

Эндрю: **Знаешь, ты права. Никогда не задумывался об этом, но ведь диаграммы burn-down – основа Agile-проектов – не так уж отличают-**

ся от освоенных объемов. Неужели они настолько эффективнее, чем скучные 50-летние методы управления проектами, применяемые при разработке, например ядерного реактора или космического челнока?

Алекс: Причина, по которой я считаю диаграммы burn-down исключительно удачным решением, состоит в том, что это понятный, общий язык для всех участников проекта. И давайте не будем забывать о заказчиках – тем, кому нужны правильные программы, созданные в срок и в рамках бюджета. Это могут быть внутренние потребители – если вы разрабатываете внутрикорпоративное ПО, продакт-менеджеры или маркетологи – если продукт нужно будет выводить на рынок, – словом, те, кто в данном процессе играет роль потребителя.

Требуется точный учет того, что необходимо сделать и что только желательно, и стоимости выполнения этих задач и представления программы пользователю; все это крайне важно для команды. (Здесь я включаю в команду и потребителя – в лучших традициях Agile.) И все это как раз хорошо решается с помощью диаграмм burn-down. Вы не сможете назвать что-то «готовым, за исключением тестирования». Если еще надо тестировать или еще что-то делать, то ничего не готово. Готово – это готово. Готово – это то, что можно завтра поставлять.

Любой код, проверенный в сборке, одобренный, принятый с любой точки зрения, должен быть непосредственно готов к поставке. Это очень важно. Поэтому значение слова *готово* в Agile-методологии более конкретно, чем в иных. И стоимость после оценки и планирования «это мы сделаем сначала, а это потом» тоже становится прозрачной. Нет никакой волшебной палочки, которая помогла бы бесплатно и сразу все это вам рассказать. Ее нет ни в одном программном проекте. Всегда существенную часть работы в проекте составляют определение того, что, собственно, сейчас делается, и попутная модификация задач под актуальные нужды заказчика или потребителя.

Эндрю: **Пожалуй, многое из того, о чем вы говорите, станет большой неожиданностью для тех, кто считает Agile синонимом к «мы не планируем наши проекты».**

Алекс: Agile-разработки и диаграммы burn-down помогают считать все факторы, которые в более консервативных, традиционных методиках вели к затягиванию сдачи проекта, частью процесса разработки ПО, и так и должно быть, потому что эти факторы либо изначально известны, либо выясняются в процессе разработки. Это имеет отношение не столько к командному аспекту, сколько к самой разработке, которую должна осуществлять команда. Так что это, на мой взгляд, две стороны одной медали.

Если считать заказчиков, продакт-менеджеров и потребителей частью команды, то можно говорить о командном духе, но я никогда не сталкивался с этим, кроме как в специализированных Agile-группах.

Эндрю: **Интересно, что вы напрямую связали диаграммы burn-down и разговор с потребителем. Диаграммы burn-down показывают, какие усилия приложены и какая часть работы выполнена. Мне понравился ваш термин «правильные программы» и то, что вы сказали о готовности проекта: когда остается работа, например только для тестеров или службы контроля качества, часто проект считают уже «готовым». Все это мне очень нравится, потому что открыв любой учебник по контролю качества — я не только о программировании, а о качестве инжиниринга вообще, даже если вернуться на 50 лет назад, к Демингу и Джурану, — то увидишь практически те же определения качества. Разве что вы применяете их в контексте Agile-разработки.**

Дженни: **Создание правильных программ, соответствие требованиям, готовность к использованию — в Agile-группе можно обойтись без этих традиционных терминов качества, но, думаю, интересно то, что они до сих пор составляют основу того, о чем вы говорите. Отсюда непосредственная связь между командой, применяющей этот подход, с ежедневными собраниями, диаграммами и доской задач, и традиционными методами контроля качества. Думаю, для большинства программистов эта связь неочевидна.**

Эндрю: **И все это оказывает большое влияние на команду — на их совместную работу, на командный дух. Есть, наверное, и другие традиционные идеи из области контроля качества, которые являются частью Agile-методик? Готовность к изменениям — важная часть Agile-процесса. Но такая готовность и уверенность ваших заказчиков в том, что они могут рассчитывать на изменения, — это и важная часть традиционного управления проектами. Как это влияет на командный дух?**

Алекс: Думаю, в данном конкретном случае помогает постоянно обогащающаяся природа большинства Agile-процессов. Если вы выполняете, например, ежемесячную итерацию для какого-то очень большого проекта или двухнедельную для более мелкого и быстрого, вы без удивления воспримете изменения, которые могут произойти в конце итерации, потому что некоторое время посвящаете взгляду назад: «А что мы сделали в последней итерации? Что было не так? Что прошло хорошо? Научились ли мы чему-то новому, что нужно применить в будущем, чтобы избежать ошибок или воспользоваться преимуществами лучших методов?» Это означает постоянную готовность к переменам, и не случайным и непредсказуемым, а запланированным. И в этом большое

преимущество. Под запланированными переменами я понимаю еженедельные собрания по четвергам или средам, а не подход «Ну, как закончим, так и закончим».

Эндрю: **И это помогает держать под контролем расписание проекта?**

Алекс: Это помогает держать темп, ритм: вы знаете, когда дедлайн, и все, что не успели протестировать до дедлайна, войдет уже в следующую итерацию. Даже в разгар работы вы знаете, что оглядываться необходимо. Зачем работать над тремя задачами, если мы уже потратили на них много сил и больше ничего не можем добавить? Что ж, они оказались сложнее, чем ожидалось, или этот чертов компьютер постоянно ломался. Таким образом, решения могут быть абсолютно разными. Одно из них – провести переоценку оставшихся задач: может быть, можно разбить одну большую задачу на несколько маленьких, чтобы не ужасаться одной огромной.

Так что в начале следующей итерации совершенно необходимо присутствие высшего руководства или других заказчиков, потому что это их шанс что-то изменить. Они должны ориентироваться в задачах этой итерации, которые могут заключаться в том, чтобы отменить или отложить внедрение определенных функций, вынести на первый план другие, ввести совершенно новые, принять решение о том, что изначально хорошая идея в данном случае, к сожалению, не годится, так что использовать ее не нужно, – и так далее, по порядку.

И опять же отмечу, что это не какое-то случайное изменение, взятое с потолка, неожиданно. Оно запланировано. Оно спрогнозировано. Вы точно не знаете, что именно изменится, но знаете, что перемены скорее будут, чем нет, потому что изменится сама бизнес-реальность.

Эндрю: **Из ваших слов я делаю вывод, что важная составляющая успешности команды, создания хорошего ПО, – не просто определение ожиданий, например руководства и потребителей, но управление ожиданиями самой команды.**

Алекс: Отличное наблюдение. Это улица с двусторонним движением. Инженеры и вообще разработчики, в особенности лидер команды, должны регулярно разговаривать с теми, кто с надеждой ждет завершения проекта.

Дженни: **Когда вы заговорили об ожиданиях команды и их соответствии ожиданиям потребителя, мне кое-что вспомнилось. Мне довелось работать в нескольких Agile-проектах, которые только из-за этих двух факторов окончились неудачей. Наши потребители, участвующие в Agile-процессе с правом решающего голоса, полностью меняли свои требования в течение нескольких итераций – до того, что при-**

ходилось переписывать все заново и уже вообще ничего не хотелось делать. Это первое.

И второе. В Agile-проектах мне попадались разработчики, которые не могли обуздать свои творческие порывы, считая необходимым добавить своего рода позолоту — функции, ненужные заказчику и, собственно, не заказанные, но интересные самим программистам. Иногда это улучшало программы и нареканий не было, но чаще приводило к серьезным проблемам с качеством.

Алекс: Да. Инженеры, особенно молодые и талантливые, но еще недостаточно битые жизнью, действительно склонны к лишней работе. Здесь я полностью согласен. Но обнаружилось, что этому хорошо препятствует нежно любимый мной метод, применявшийся в Google еще до моего прихода. Ранее я пытался внедрить его, но это трудно сделать, если такой метод не является уже частью корпоративной культуры, — это обязательные проверки кода. Я поручил это своему главному сотруднику — Крэйгу Сильверстейну. В базу кода не попадает ни один сегмент, который не был бы проверен и одобрен другим членом команды. Это не та структурированная проверка кода, которую имели в виду в 1970-е и которая требовала многих часов подготовки и проектирования, доски и прочего.

Это абсолютно простой процесс, прекрасно выполняемый по электронной почте или через Сеть с помощью несложных инструментов. Смысл здесь в том, что если вы в курсе всех изменений и всех проверок кода, участвовали вы в них или нет, то не допустите ошибку из-за того, что какое-то изменение сказывается на базе кода. Я говорю здесь о проверках внутренних, осуществляемых самой командой, например в случае с позолотой для программ, который вы упоминали, в отличие от изменений, которые требуют потребители (это уже другой случай).

Эндрю: **Я вполне понимаю, как такие проверки кода помогают решить вторую проблему, о которой говорила Дженни, — проблему инженерной позолоты. А как насчет первой проблемы, с которой я тоже сталкивался? Хорошо помню тот случай, о котором она говорила: ее команду погубили бизнесмены-заказчики, которые требовали в первую очередь того, что никогда не должно иметь первостепенного значения. Потребитель полностью и неожиданно изменил направление работы, в итоге пришлось практически переписывать всю базу кода с нуля. Уверен, что такая ситуация знакома не только ей. Есть ли хороший Agile-метод, решающий эту проблему?**

Алекс: Здесь мы подходим к двум главным стремлениям команды разработчиков: делать правильные программы и делать программы правильно. Главным образом мы с вами говорили о том, как делать про-

грамму правильно. Учитывая, что программа должна делать то-то и то-то, сделай ее именно такой. Красивой, надежной, эффективной, безошибочной и так далее.

Но, возможно, важнее сделать правильную программу: программу с таким набором функций, который способствует успеху компании. Понятно, что если вы никогда не принимали точку зрения бизнесменов, то может показаться, что все перемены, эти попросту абсурдные радикальные изменения, которые вас просят внести, – просто каприз, произвол. Но, поверьте, смысл у них есть, разве что в вашей компании полный раздрай, а в этом случае вам остается только распечатать резюме на хорошем лазерном принтере.

Если компания функционирует нормально, попробуйте поставить себя на место бизнесмена. Они изменяют мир под себя, и только это помогает им держаться на плаву.

Большинство школ Agile-программирования придерживаются правила, которое я не вполне одобряю, – избегать универсальности. Универсальность порой требует усилий, но иногда она помогает сберечь время. Простейший пример: узкоспециальная программа, вычисляющая остаток от деления на 3, может оказаться дороже программы, которая вычисляет остаток от деления на любое число и которой можно передать 3 как аргумент.

Так что если у вас хорошее чутье – чутье проектировщика и, что не менее важно, исполнителя, для которого универсальность не только ничего не стоит, но и позволяет удешевить работу, – то вы способны адекватно воспринять изменения и, будучи готовыми к ним, создавать классы, модули, пакеты и шаблоны, которые можно конфигурировать для различных ситуаций.

Просто если большую часть вашей работы можно начинать не с чистого листа, то мешать этому может только досадное недоразумение. Если имеется код для повторного использования, то его и правда стоит задействовать. А если бизнесмены вашей фирмы действительно чересчур капризны и только и делают, что тратят ваше время и деньги компании, то единственный выход – лазерный принтер. Но скорее всего они просто пытаются делать все от них зависящее, чтобы выжить в этом суровом, изменчивом и полном неожиданностей мире.

Эндрю: **Наш разговор об Agile-методах напомнил мне то, что случилось несколько лет назад. Нас с Дженни пригласили в известную компанию с лекцией об улучшении разработки ПО. В конце лекции разгорелась дискуссия об Agile-разработке, в то время достаточно новом явлении. И тут встает разработчик, очевидно работавший там уже давно. Здоровый мужик с большой седой бородой – на вид типичный**

программист. Остальные к нему явно прислушивались. И вот он поднимается и... Дженни, помнишь, что он сказал?

Дженни: **Он сказал: «Agile означает, что ты ничего не записываешь, а просто начинаешь программировать».**

Алекс: Да. У меня есть очень известный коллега, который аплодировал бы словам этого парня. Это популярный блогер Steve Yegge (работает в Сиэтле). Мы с ним, скажем так, несколько расходимся в технических вопросах.

Если ввести в блогосфере запрос «Steve Yegge agile», то найдется миллион постов, в которых он порицает саму идею планирования и объясняет, что единственный правильный способ – примерно такой, как сказал ваш разработчик. Просто начинай программировать – и все само собой наладится. Не ограничивайтесь моим конспектом его идей, прочитайте его собственные посты. Они отлично написаны! И если быть до конца честным, сыграв адвоката дьявола, то есть случаи, когда его подход сработал. Мой любимый пример... Знаете такую программистскую компанию Autodesk?

Эндрю: **Конечно.**

Алекс: Никогда не интересовались, при чем тут рабочий стол (desk)?

Как-то они решили сделать приложение для офисной автоматизации. Автоматизировать рабочий стол – еще до появления Microsoft Office и других интегрированных пакетов. А двое сотрудников на весьма ответственных инженерных должностях в стартапе, которые не приветствовали эту идею, в собственное свободное время вместо нее разрабатывали приложение для автоматизации проектирования.

Эндрю: **И так они создали AutoCAD?**

Алекс: Да. Они решили, что это поинтереснее программ для обработки счетов-фактур и прочего, для которых, по мнению их бизнесменов, были огромные рынки. В итоге приложение AutoDesk так и не увидело свет, зато AutoCAD покорил мир. Компания не сменила название, но несмотря на это процветает в области автоматизированного проектирования уже больше 20 лет. Есть отличная книга одного из основателей, где излагается вся история Autodesk. Так что это контрпример, когда те, кто делал программы в соответствии с общей политикой компании, как выяснилось позже, так и не создали ничего приемлемого. А бунтари, которые рискнули заняться чем-то совершенно иным и «потратили» ресурсы компании на то, что не вписывалось в бизнес-модель фирмы... на деле спасли компанию и обеспечили ей огромный успех.

Но это практически история о человеке, который выиграл 137 миллионов долларов в лотерею. Так бывает, но рассчитывать на это я бы не стал.

Эндрю: **Это чем-то принципиально отличается от проектов типа skunkworks[1?]**

Алекс: По моему мнению, проекты skunkworks – это то, что высшее руководство на деле должно поддерживать. Команда не контактирует с остальными сотрудниками, особенно со средним звеном. Основная роль среднего звена в компании – предотвращать любые изменения (по крайней мере, когда имеешь с ними дело, то так кажется). Хорошие опытные инженеры и, как ни странно, некоторые высокопоставленные руководители часто обладают хорошей интуицией и могут позволить кому-то там в углу заниматься своими делами. Этот же проект, как я могу понять из опубликованной истории Autodesk, полностью был решением группы инженеров.

Другой пример – Microsoft Internet Explorer 1, первый выпуск. Некоторые инженеры считали, что новая программа для Интернета необходима, что Microsoft должен выпустить броузер, но... длинная цепь руководителей среднего звена прикончила эту дурацкую идею. Никто ломаного гроша не давал за броузер. В итоге парни собрались в углу и решили все равно писать броузер. Как вдруг генеральный директор Билл Гейтс выпустил меморандум, в котором говорилось, что Интернет – это все, это будущее компании, что все должно с этого времени измениться. И ребята смогли практически на следующий день представить практически работоспособную программу.

Это можно назвать проектом skunkworks, но и здесь я бы не ставил на это. Если вы работаете в неправильно функционирующей компании, то, возможно, это единственный способ выжить. Но ведь бывают и другие компании. Если все в фирме движутся в одном направлении, все заинтересованы в том, чтобы потребители получали хорошие программы и хорошо за это платили, то нет необходимости скрываться и делать что-то свое, вместо того чтобы идти в одной упряжке с остальными.

Дженни: **Полагаете, в таких условиях лучше сработает более структурированный и более традиционный Agile-подход?**

---

[1] Проект типа skunkworks – обычно технологический проект, разрабатываемый небольшой автономной группой, в основном, с целью инноваций. Название происходит от Skunk works — секретного отделения фирмы Lockheed Martin, организованного в 1943 г. для разработки реактивного истребителя XP-80. – *Прим. ред.*

Алекс: Да, полагаю, потому что он спаян с идеей изменений, чем выгодно отличается от других хороших методов – например от унифицированного процесса Rational (Rational Unified Process, RUP). Постоянные итерации и идея установления для них определенного времени, фиксированных дат и так далее для RUP являются факультативными: это можно сделать, но не обязательно; не они составляют основную идею процесса. Главная идея – это использование огромного количества диаграмм, которые не может прочитать никто, кроме «касты» ваших экспертов. Думаю, многим командам нравится исписывать огромные кипы бумаги, потому что ими можно как стеной отгородиться от остальных отделов организации, которые иначе могли бы потребовать отчетности и прозрачности. Это, конечно, немного утрированно, но вы поняли, о чем я.

Идея создать бумажную стену перед началом кодирования – о! Это как раз тот самый раздрай, вызвавший столь резкую реакцию у программиста, которого вы процитировали: «От нас требуют слишком много тупых спецификаций, так давайте ничего не будем писать». Лучшее решение, как всегда, где-то между этими крайностями.

21

Команды и инструменты

Карл Фогель

Эта глава[1] посвящена преобразующему эффекту, который хорошие инструменты могут оказать на способность команды к совместной работе.

Вспомните, что много лет Всемирная паутина состояла главным образом из статических страниц, работа с которыми требовала технической квалификации, при этом читатели не могли взаимодействовать с их содержанием; разве что можно было послать электронное письмо по соответствующему адресу *webmaster@*. Затем несколько прозорливцев начали выпускать программы, позволяющие любому пользователю с базовыми компьютерными навыками ввести текст в Сеть, и программы, которые давали читателям возможность оставлять комментарии или даже самостоятельно редактировать эти страницы. Ни блоги, ни вики-ресурсы не были результатом *технической* революции; как и факсовый аппарат, их можно было придумать еще за несколько лет до этого – если бы кому-то эта идея вообще пришла в голову. Но их появление позволило людям скорее самоорганизоваться в продуктивные сети.

В этой главе вы найдете три истории о том, как наличие (или отсутствие) хороших инструментов повлияло на команду. Те инструменты, о которых здесь говорится, обладают значительно более узкими возможностями, чем блоги и вики, но сама их специфичность выявляет некоторые принципы инструментов для совместной работы. Поскольку мой опыт главным образом связан с работой над проектами с открытым

[1] Глава издается по лицензии Creative Commons Attribution 3.0 (*http://creativecommons.org/licenses/by/3.0/*).

исходным кодом, я остановлюсь именно на них, но те же принципы, скорее всего, применимы и к любой совместной работе.

Как работают проекты с открытым исходным кодом

Если вам уже доводилось участвовать в проекте с открытым исходным кодом, можете пропустить эту часть. Если же нет, то вот основы работы в таких проектах, без которых вряд ли будет понятен смысл работы с инструментами, обсуждаемыми в этой главе.

Программы с открытым исходным кодом[1] – это программы, которые выпущены со свободными авторскими правами и могут быть скопированы, изменены, использованы и переданы далее (в измененной или не измененной форме). Большая часть программ, на которых держатся Интернет и Всемирная паутина, имеет открытый исходный код, как и постоянно увеличивающееся количество приложений для персональных компьютеров.

Программы с открытым исходным кодом чаще всего поддерживаются свободно организованными союзами разработчиков. Некоторые работают бесплатно, труд других оплачивается корпорациями, заинтересованными в поддержке этих программ. Поскольку участники этих союзов часто разбросаны по часовым поясам и, возможно, даже никогда друг друга не видят, проекты в очень большой степени зависят от изощренных инструментов совместной работы, систем отслеживания ошибок, листов рассылки и архивов, сетевых чатов, репозиториев версий, вики и так далее. Знать все эти инструменты необязательно; я расскажу только о тех, которые будут упоминаться далее.

Каждый проект с открытым исходным кодом нужно как-то организовать и впоследствии управлять им. Обычно это решается неформальным образом: проекты обычно начинает небольшая группа (иногда даже кто-то один), а через некоторое время, когда появляются заинтересованные добровольцы и приступают к работе, основатели проекта решают, что будет неплохо пригласить кого-то из них и на роль основных специалистов[2], Но поскольку в проекте уже много участников, то

---

[1] Более старый термин *бесплатные программы* – синоним. Здесь недостаточно места, чтобы объяснить, почему одно и то же называется по-разному. Более подробную информацию вы найдете на сайте *http://producingoss.com/en/introduction.html#free-vs-open-source*.

[2] Возможно, вы удивились: зачем нужна «основная группа», если доступ к исходному коду открыт всем? Не значит ли это, что основным специали-

может быть трудно отслеживать их всех и разбираться, кто их них достоин повышения. Первый инструмент, который мы рассмотрим, появился, чтобы решить подобную проблему, возникшую в одном проекте.

Contribulyzer

Как и большинство проектов с открытым исходным кодом, Subversion[1] постоянно пытается подобрать новых ведущих разработчиков. Собственно, одна из основных задач текущей основной группы состоит в том, чтобы анализировать код участников проекта, решая, кого стоит пригласить в основную группу. Чтобы беспристрастно оценить слабые и сильные стороны кандидатов, мы (ведущие разработчики) создали закрытую рассылку – одну из немногих закрытых в проекте. Когда кто-то считает, что какой-то участник достоин повышения, его кандидатура предлагается в рассылке, затем поступают отзывы. Мы тщательно проверяем каждого кандидата, поскольку хотим достичь полного согласия, прежде чем сделать ему предложение: отзывать предложение было бы нелепо, и мы не хотим создать подобную ситуацию.

Эта негласная проверка сложнее, чем кажется. Часто патчи[2] одного и того же участника рассматриваются разными специалистами в разных ситуациях, а это значит, что никто из основной группы не может

стом становится любой желающий? Не совсем так. В таких проектах все могут сделать копию кода и сделать с этой копией что угодно. Но когда люди собираются в группу и договариваются что-то сделать с одной определенной копией вместе, они, разумеется, контролируют процесс вступления в группу. Например, любой может перевести Библию для собственного удовольствия, поскольку она находится в общей собственности, но никто не может заставить одну группу переводчиков работать вместе с другой; а если обе эти группы не работают вместе, то в результате получатся два независимых перевода. В среде программистов с открытым исходным кодом такая ситуация называется *вилкой*, как развилка на дороге: две независимых копии с постоянно увеличивающимся набором расхождений. Но на практике подобные вилки встречаются редко: чаще программисты концентрируют усилия на какой-то одной копии.

[1] Subversion – система управления версиями с открытым исходным кодом, см. *http://subversion.tigris.org/*.

[2] *Патч* – кусок кода, представляющий собой исправление ошибки. Отсылается в специальном патчевом формате. Детали этого формата здесь значения не имеют, так что мы будем рассматривать патч как предложение модификации программы, поданное в максимально подробной форме – вплоть до переписки отдельных строк кода с указанием причин.

представить полную картину всех действий участника. Даже когда патчи одного и того же участника поступают к одному и тому же ведущему разработчику (сознательно или случайно), они могут сдаваться нерегулярно – раз в несколько месяцев или год, так что специалисту трудно бывает оценить общее качество кода, отчеты об ошибках, предложения по проекту и так далее.

Я впервые осознал стоящую перед нами проблему, заметив, что всплывавшие в закрытой рассылке имена получают отклик либо запоздалый, либо вовсе *никакого*. Это казалось необычным, поскольку предложенные кандидатуры активно участвовали в проекте, чаще всего относительно недавно, и некоторые патчи уже были приняты, подчас после нескольких итераций проверок и дискуссий на открытых листах рассылки. Однако вскоре выяснилось, что происходит: специалисты не решались полагаться только на свою память, определяя, что успел сделать кандидат (никому не хочется поддерживать кого-то, кто потом окажется придурком), но в то же время их не радовала и перспектива копаться в архивах рассылки и истории изменения кода, чтобы эту память освежить. Я чувствовал, что многие из нас пришли к классической модели поведения типа «Так, значит Такого-то предлагают в основную группу. Ладно, подожду с отзывом до выходных, когда у меня будет пара часов порыться в архивах и посмотреть, что он там делал». Разумеется, в итоге по ряду причин «пары часов» в его распоряжении в эти выходные не оказывается, все откладывается снова и снова... Тем временем кандидат понятия не имеет, что происходит, и продолжает разрабатывать патчи, в то время как давно мог бы участвовать напрямую. Это означает, что специалисты из основной группы вынуждены просматривать новые патчи, в то время как, приняв кандидата в группу, могли убить двух зайцев: ему бы не требовалась помощь для введения патчей в код, к тому же он мог бы работать и с патчами других участников.

Наблюдения за самим собой подтверждали эту гипотезу: знакомое чувство раздражения появлялось каждый раз, когда нужно было решить вопрос по новому кандидату, и не потому, что я не хотел сделать его ведущим разработчиком, а из-за того, что не знал, где взять время на ответственное заключение по этому предложению.

Наконец, как-то вечером я отвлекся от текущей работы и стал искать решение. Первая идея была далека от оптимальной и не автоматизировала окончательно задачу подбора информации, необходимой для оценки участника. Но даже частичная автоматизация существенно сокращала время оценки, и этого было достаточно, чтобы, как говорится, перестать буксовать. Со времени установки этой системы предложения по новым кандидатам почти всегда получали своевременные отклики

благодаря почерпнутой с помощью этой системы информации, поскольку теперь не нужно стало копаться в архивах, убивая на это кучу времени. Новая система взяла на себя одни и те же рутинные операции, которые до того каждый специалист должен был выполнять самостоятельно; теперь же они производились *один раз*, и их результаты сохранялись для дальнейшего использования.

Эта система называется Contribulyzer (*http://www.red-bean.com/svnproject/contribulyzer/*): она отслеживает все действия участников и записывает их на одной веб-странице. Если разработчики из основной группы хотят узнать, готов ли вступить в их ряды данный участник, им достаточно взглянуть на соответствующую страницу Contribulyzer, где говорится об этом участнике, рассмотреть его действия в общем, а затем при необходимости сосредоточиться на деталях.

«Но каким образом компьютерная программа „отслеживает“ то, что делает участник? Это какая-то подозрительная дьявольщина», – скажете, возможно вы. Никакого волшебства: только небольшое вмешательство человека – и готово. Мы увидим, как это работает. Но сначала посмотрим на результаты. На рис. 21.1 изображена главная страница нашего сайта Contribulyzer.

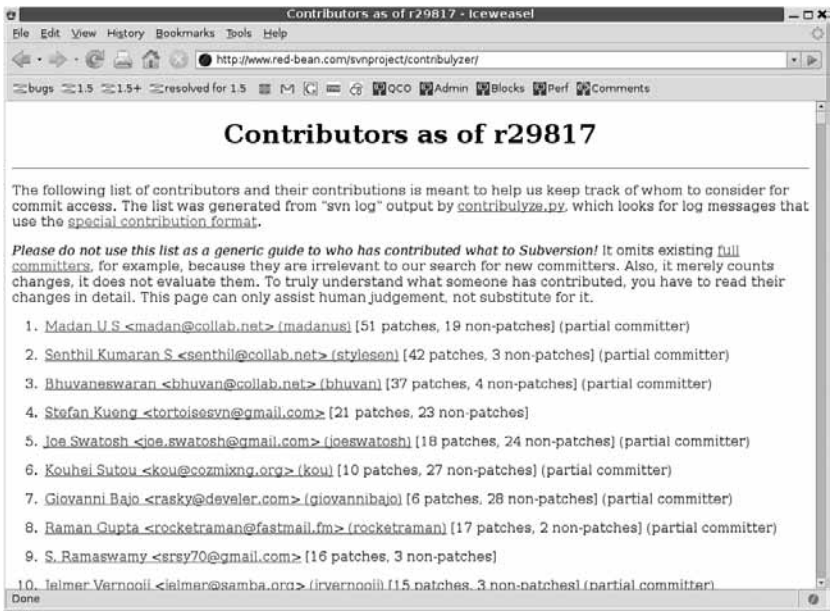


Рис. 21.1. *Главная страница Contribulyzer*

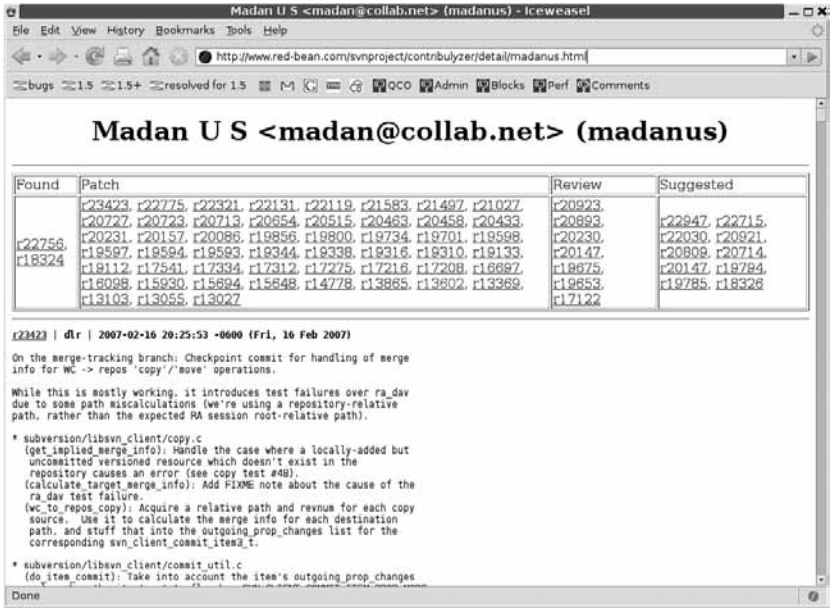


Рис. 21.2. *Страница участника в Contribulyzer*

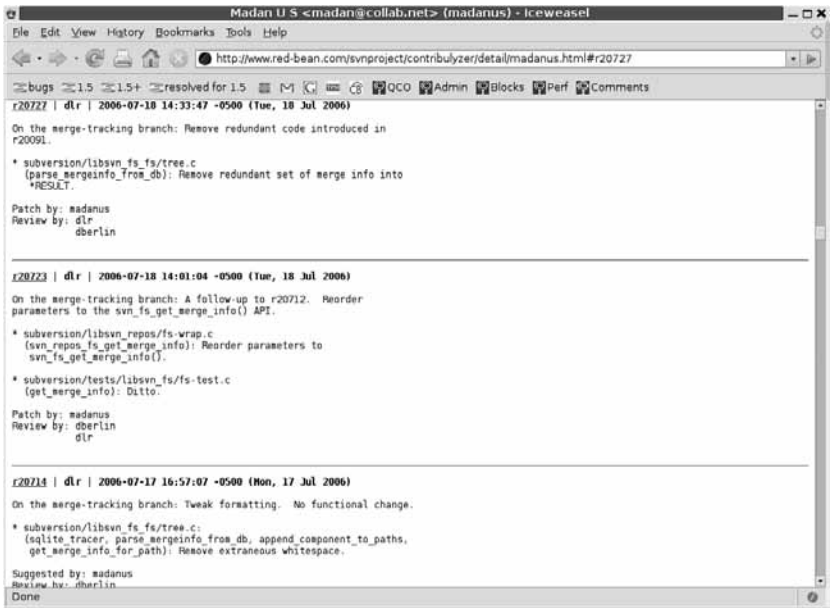


Рис. 21.3. *Страница изменения в Contribulyzer*

Щелкнув на имени участника, вы попадете на страницу, где указано, что именно делал данный участник. На рис. 21.2 показана верхняя часть страницы участника Ю. С. Мадана.

Четыре категории в верхней части страницы указывают, в чем именно Мадан принимал участие. Каждый индивидуальный вклад представлен *ревизионным номером (revision number)* – номером (начинающимся с *r*), уникальным для данного конкретного изменения. По ревизионному номеру можно запросить детали изменения в центральном репозитории (узнать, какие именно строки кода были изменены и как это было сделано). В случаях *r*22756 и *r*18324 Мадан нашел ошибки, которые и были исправлены. Но большей частью это были патчи: он написал изменение, которое затем подтвердил кто-то из ведущих разработчиков. В остальных случаях он либо проверял патч, примененный кем-то еще, либо предлагал исправление, которое по какой-то причине не было применено.

Эти четыре столбца уже дают хорошее представление о деятельности Мадана. Более того, каждый номер изменения является гиперссылкой, ведущей к его краткому описанию. Это описание называется *сообщением лога (log message)*: это краткое описание изменения с указанием его причин. В репозитории это сообщение хранится вместе с изменением; это ключевой ресурс для тех, кто позднее захочет понять смысл изменения.

Если щелкнуть на r20727, в верхней части следующего экрана отобразится сообщение лога для этого изменения, как показано на рис. 21.3.

Номер изменения здесь тоже является ссылкой, но на этот раз на страницу подробного описания самого изменения, которую можно просмотреть с помощью броузера репозитория ViewVC (*http://www.viewvc.org/*), как на рис. 21.4.

Здесь можно узнать, какие именно файлы были изменены, а если щелкнуть на modified, то вы увидите и само изменение кода, что показано на рис. 21.5.

Как видите, расположение информации в Contribulyzer соответствует тому, что нам нужно держать в памяти для оценки работы участника проекта. Здесь можно увидеть в целом, какой вклад внес данный сотрудник, точный отчет о каждом его действии, детальное описание всего пути, который проделал его код. Но как вся эта информация попадает в Contribulyzer?

Рис. 21.4. *Страница изменения в броузере ViewVC*

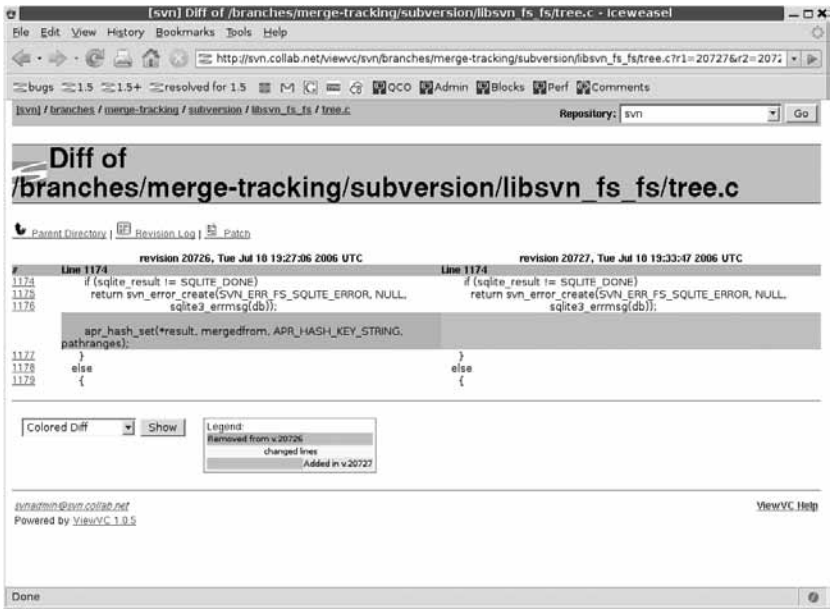


Рис. 21.5. *Страница сравнения файлов в броузере ViewVC*

The Catch

К сожалению, Contribulyzer – это не какая-то чудесная программа с искусственным интеллектом. Единственная причина того, что ей известно, кто и какие изменения внес, – мы *рассказали* это ей. И хитростей здесь две:

- Уменьшить затраты как только возможно.
- Убедить людей на конкретных примерах, что затраты окупятся.

Первое условие выполнить легко: Contribulyzer берет данные из сообщений лога Subversion, которые поступают при каждом изменении. У нас всегда есть некоторые условия написания изменений, например именовать каждый элемент кода, которого коснулись изменения. Поддержка Contribulyzer – это добавление всего одного условия, стандартный способ атрибуции изменений, идущих не от членов основной группы, которые следят за изменениями в репозитории.

Стандарт прост. Сначала идет глагол (для каждого предполагаемого типа изменений), за ним предлог by, а после него – имена участников, приложивших руку к этому изменению. Чаще всего у изменения только один автор, но если их несколько, то они указываются в следующих строках:

```
Patch by: имя_1_возможно_с_E-mail
          имя_2_возможно_с_E-mail
Found by: имя_3_возможно_с_E-mail
Review by: и так далее...
Suggested by: и так далее...
```

(Эти условия подробно описаны на сайте *http://subversion.tigris.org/hacking.html#crediting*.)

Одна из причин, по которой удалось быстро убедить всех перейти на новый стандарт, – писать сообщения лога стало проще. Прежде мы общались с участниками в совершенно разных выражениях: каждый раз, когда мы применяли предложенный участником код, приходилось думать, как описать изменение. Например, один раз это могло быть так:

```
Уберите лишний код в r20091. Патч предложил имя_1_возможно_с_E-mail.
```

А второй раз – так:

```
Исправьте ошибку в передаче маркера. (Спасибо за патч такому-то.)
```

И хотя новому условию пришлось учиться, *однажды выученное* сэкономило усилия: теперь не приходилось думать над строением фразы, поскольку у нас был Единый Стандарт.

Но ввести новый стандарт в проект не всегда просто. И это можно существенно облегчить, если выполнить и второе условие, то есть показать выгоду *прежде*, чем предлагать всем чем-то поступиться. К счастью, нам это удалось. Сообщения лога Subversion можно изменять (в отличие от некоторых других систем управления версиями, где ничего менять нельзя). Это значило, что после написания кода Contribulyzer, который обрабатывал сообщения лога, отформатированные по новому стандарту, мы могли вернуться к уже имевшимся логам проекта и переделать их в соответствии с этим стандартом, а затем постфактум создать страницу Contribulyzer, на которой отражалась вся история проекта. Этого мы добились в два этапа: сначала нашли все символы @ в сообщениях лога, чтобы обнаружить фрагменты, где упоминался чей-то электронный адрес (поскольку, упоминая участников, мы часто приводили их электронные адреса), а затем искали по именам, выявленным в результате первого поиска, но уже без адресов. Получившийся список сообщений лога – около тысячи штук – мы с помощью нескольких добровольцев и одного жутко сложного макроса всего за одну ночь привели к новому формату.

Таким образом, предложению нового стандарта сопутствовала демонстрация того, что он может для нас сделать: страницы Contribulyzer были готовы к работе с того момента, как Contribulyzer был представлен команде. Это сразу же показало его преимущества, и все поняли, что новые требования по форматированию сообщений лога – сравнительно небольшая цена.

Недостатки Contribulyzer

У разработчиков открытого кода есть поговорка, давно уже популярная и за пределами их круга:

> Лучшее не должно быть врагом хорошего.

Contribulyzer может делать значительно больше, чем умеет сейчас. Это только шаг к совершенной системе отслеживания деятельности. В идеальном мире такая система должна собирать информацию с помощью архивов рассылки и системы отслеживания ошибок, а не только с помощью системы управления изменениями. Мы бы могли тогда переходить с сообщения лога, где упоминается участник, к ветви дискуссии листа рассылки, где участник обсуждает свои изменения с другими разработчиками, – и наоборот, то есть переходить от ветви дискуссии в рассылке к собственно действию. Точно так же мы могли бы вести статистику сообщений об ошибках данного участника, приведших к изменениям или нетривиальной ветви дискуссии (что может сказать нам, что к сообщениям этого участника об ошибках нужно впредь относиться внимательнее из-за их очевидной эффективности).

Нашей целью было не создание рейтинговой системы; это было бы бесполезно и, пожалуй, даже вредно, потому что создавало бы излишнее давление и сводило оценку к чисто количественному сравнению участников. Скорее целью было облегчить поиск информации о человеке, *который, как уже выяснилось, вас интересует*.

Любой участник проекта с открытым исходным кодом оставляет след. Даже вопрос на листе рассылки оставляет след – по меньшей мере одно сообщение, а может и больше, если дискуссия им не ограничивается. Но на данный момент эти следы практически незримы: нужно вручную прочесать множество архивов, баз данных, систем управления версиями, чтобы составить относительно законченную картину деятельности данного субъекта.

Contribulyzer – это лишь шаг в направлении автоматизации обнаружения таких следов. Я включил рассказ о нем в эту главу, чтобы показать, как даже небольшая автоматизация способна внести существенные изменения в способность команды к совместной работе. Хотя Contribulyzer работает только с логами управления изменениями, он все равно помогает сберечь много времени и сил, потому что сообщения лога часто содержат отсылки к нужным ярлыкам системы отслеживания ошибок и веткам дискуссий листов рассылки, так что, раз мы можем быстро получить сообщения лога, то битва наполовину выиграна.

Конечно, я далек от идеализации Contribulyzer: разумеется, есть много аспектов организации проекта с открытым исходным кодом, которых он не касается. Но он значительно уменьшает объем работ по оценке новых потенциальных сотрудников основной группы, и таким образом наши оценки становятся более точными. Для однодневного изобретения не так уж и плохо.

Написание метакода редко может сравниться по продуктивности непосредственно с кодированием, но обычно игра стоит свеч. Если вы правильно определили проблему и увидели ясное техническое решение, то одноразовое усилие может приносить устойчивый результат в течение всей жизни проекта.

Коммит-письма и помехи в работе

Следующий пример показывает, что может случиться, если не уделить достаточное внимание использованию инструментов. Простейшее, на первый взгляд, интерфейсное решение может сильно изменить поведение людей. Для начала введу вас в курс дела.

Большинство проектов с открытым исходным кодом используют *коммит-рассылку (commit email list)*. По рассылке письмо приходит каж-

дый раз, когда в основном репозитории происходят изменения, – это письмо автоматически генерируется репозиторием. Обычно в каждом письме указываются автор и время изменения, соответствующее сообщение лога и само изменение шаг за шагом (в формате «патча» (patch), как было указано выше, только в данном случае по историческим причинам патч называется «диффом» (diff), или сравнением версий). Письмо также может содержать URL-адреса – постоянные ссылки на все изменение или его составные части.

Типичный коммит проекта Subversion:

```
From: dionisos@tigris.org
Subject: svn commit: r30009 - trunk/subversion/libsvn_wc
To: svn@subversion.tigris.org
Date: Sat, 22 Mar 2008 13:54:38 -0700

Author: dionisos
Date: Sat Mar 22 13:54:37 2008
New Revision: 30009

Log:
Fix issue #3135 (property update on locally deleted file breaks WC).

* subversion/libsvn_wc/update_editor.c (merge_file): Only fill WC file
   related entry cache-fields if the cache will serve any use (ie when
   the entry is schedule-normal).

Modified:
   trunk/subversion/libsvn_wc/update_editor.c

Modified: trunk/subversion/libsvn_wc/update_editor.c
URL: http://svn.collab.net/viewvc/svn/trunk/subversion/libsvn_wc/
     update_editor.c?pathrev=30009&r1=30008&r2=30009
==============================================================================
--- trunk/subversion/libsvn_wc/update_editor.c Sat Mar 22
    07:33:07 2008 (r30008)
+++ trunk/subversion/libsvn_wc/update_editor.c Sat Mar 22
    13:54:37 2008 (r30009)
@@ -2621,8 +2621,10 @@ merge_file(svn_wc_notify_state_t *conten
   SVN_ERR(svn_wc__loggy_entry_modify(&log_accum, adm_access,
           fb->path, &tmp_entry, flags, pool));

-  /* Log commands to handle text-timestamp and working-size */
-  if (!is_locally_modified)
+  /* Log commands to handle text-timestamp and working-size,
+     if the file is - or will be - unmodified and schedule-normal */
+  if (!is_locally_modified &&
+      (fb->added || entry->schedule == svn_wc_schedule_normal))
```

```
{
  /* Adjust working copy file unless this file is an allowed
     obstruction. */
```

При правильном подходе коммит-письма – очень полезные инструменты совместной работы в программных проектах. Это идеальное сочетание автоматизированной информации и человеческого разума. Каждое изменение попадает в почтовый ящик разработчика в виде понятной и удобной единицы – электронного письма. Разработчик может следить за изменениями с помощью удобного и знакомого интерфейса (почтового клиента), и если что-то кажется ему подозрительным, то он может ответить, процитировав лишь интересующие его части изменения, и ответ этот автоматически будет помещен в ветвь дискуссии, которая связывает изменение с комментариями всех заинтересованных лиц. Таким образом, применяя уже имеющиеся возможности управления данными, то есть электронную почту, участники, а также другие программы могут с удобством отслеживать последствия любого изменения[1].

Но в другом моем проекте GNU Emacs[2] дела обстоят несколько иначе. Отчасти по историческим причинам, отчасти из-за особенностей тамошней системы управления версиями[3] каждый коммит в GNU Emacs порождает *два* коммит-письма: одно содержит сообщение лога, а другое – сам дифф.

Письмо с сообщением лога выглядит примерно так:

```
From: Juanma Barranquero <lekktu@gmail.com>
Subject: [Emacs-commit] emacs/lisp info.el
To: emacs-commit@gnu.org
Date: Sat, 08 Mar 2008 00:09:29 +0000

CVSROOT:    /cvsroot/emacs
Module name:    emacs
Changes by:    Juanma Barranquero <lektu>    08/03/08 00:09:29

Modified files:
    lisp           : info.el

Log message:
```

[1] См. *http://producingoss.com/en/vc.html#commit-emails* и *http://producingoss.com/en/setting-tone.html#code-review*.

[2] GNU Emacs – программа для редактирования текста, которую предпочитают многие программисты, одна из старейших бесплатных постоянно поддерживаемых программ. См. *http://www.gnu.org/software/emacs/*.

[3] CVS – см. *http://www.nongnu.org/cvs/*.

```
    (bookmark-make-name-function, bookmark-get-bookmark-record):
    Pacify byte-compiler.

CVSWeb URLs:
http://cvs.savannah.gnu.org/viewcvs/emacs/lisp/info.el?cvsroot=
       emacs&r1=1.519&r2=1.520
```

А письмо с диффом выглядит так:

```
From: Juanma Barranquero <lekktu@gmail.com>
Subject: [Emacs-diffs] Changes to emacs/lisp/info.el,v
To: emacs-diffs@gnu.org
Date: Sat, 08 Mar 2008 00:09:29 +0000

CVSROOT:    /cvsroot/emacs
Module name:    emacs
Changes by:    Juanma Barranquero <lektu>    08/03/08 00:09:29

Index: info.el
===================================================================
RCS file: /cvsroot/emacs/emacs/lisp/info.el,v
retrieving revision 1.519
retrieving revision 1.520
diff -u -b -r1.519 -r1.520
--- info.el    7 Mar 2008 19:31:59 -0000    1.519
+++ info.el    8 Mar 2008 00:09:29 -0000    1.520
@@ -3375,6 +3375,8 @@

 (defvar tool-bar-map)
 (defvar bookmark-make-record-function)
+(defvar bookmark-make-name-function)
+(declare-function bookmark-get-bookmark-record "bookmark" (bookmark))

 ;; Autoload cookie needed by desktop.el
 ;;;###autoload
```

Вместе эти письма содержат ту же информацию, что и одно письмо в проекте Subversion, проиллюстрированное выше. Но ключевое слово тут – *вместе*. А ведь приходят они не вместе, а раздельно. Хотя программист внес только одно изменение[1], не создается единое письмо, содержащее все, что нужно для понимания и проверки этого изменения. Для проверки изменения нужно видеть сообщение лога, чтобы понять,

[1] Некоторые предпочитают термин *набор изменений (changeset)* для описываемого мною случая – ситуации, когда, возможно, изменены несколько файлов, однако эти изменения входят в одну логическую группу. Можно говорить об изменении или наборе изменений, в данном контексте это одно и то же.

зачем изменение было произведено, и дифф, чтобы увидеть, удовлетворяет ли имеющийся код этому намерению.

Получается, что если не видишь сразу оба эти сообщения, проверку осуществить затруднительно.

Или так кажется, судя по моему очень строгому[1] исследованию, в котором я сравнивал оба проекта. В феврале 2008 года на листе разработки Subversion было 207 уникальных ветвей дискуссии и 908 сообщений. Из этих ветвей 50 были образованы коммит-письмами и ответами на них. Таким образом, 24% внимания рассылки разработки уделяется проверке коммитов (или, если считать не ветви дискуссии, а письма, то чуть больше 5%). Заметим, что ответы на коммит-письма в Subversion автоматически отправляются в главную рассылку разработки с заголовком «Ответить», то есть работать с рассылкой разработки удобно.

В то же время в феврале 2008 года рассылка разработки Emacs содержала 491 уникальную ветвь дискуссии (и 3158 писем), из которых, однако, ни одно письмо не было письмом проверки.

Ошеломленный таким результатом, я перенастроил фильтры для обнаружения писем проверки и снова провел поиск. На этот раз я нашел не более 49 писем. Но из этих 49 большинство представляли собой проверки патчей, поступивших в рассылку извне, а не проверки, основанные на коммит-письмах: только два письма действительно были письмами проверки. Но даже если посчитать все 49 (что уже слишком большое допущение), то это всего 10% трафика рассылки (а если считать письма, а не ветви дискуссии, то и вовсе 1,5%). Поскольку Emacs автоматически не перенаправляет коммит-письма в основную рассылку разработки, я задал поиск и по архивам рассылок коммитов и диффов. В феврале я не нашел ни одного случая, а в марте – два, и оба – от меня.

Итак, в двух проектах различались рейтинг коммитов и уровни трафика в рассылках разработки. Но это можно частично контролировать, если подойти к проблеме с другой стороны: какой процент коммитов подвергается проверке? В феврале 2008 года в Subversion было 274 уникальных коммита, а в Emacs 807.[2] Таким образом, в Subversion это около 18%, а в Emacs – до 6% (даже, вероятно, ниже, около 0,2%, если настоящих писем проверки действительно всего два).

---

[1] Читай «совершенно неофициальному».

[2] Если захотите сравнить эти данные с первоисточниками, помните, что при подсчете коммитов в Emacs не следует учитывать те из них, которые затрагивают только файлы *ChangeLog*, потому что (из-за некоторых странностей системы управления версиями проекта Emacs) это дубликаты других коммитов.

Что-то здесь есть, что-то в одном проекте стимулирует экспертную оценку на порядок больше, чем в другом. В чем же дело?

Я, конечно, не могу ничего доказать, но думаю, что дело как раз в том факте, что каждый коммит в Emacs создает два сообщения: одно – для лога, другое – для диффа. С помощью нормального почтового клиента без дополнительных настроек нельзя в одно и то же время увидеть сообщение лога и сообщение диффа. Таким образом, *нет приемлемого способа проверить изменение*. Не то чтобы проверка вообще невозможна или даже сложна. При желании я мог бы проверить все коммиты Emacs, как и другие разработчики. Но каждая проверка требует постоянно переключаться с одного письма на другое или переходить по URL в более коротком письме и ждать, пока загрузится страница. Это, и правда, слишком хлопотно, так что вряд ли я стану утруждаться. Да и никто не станет.

И это отрезвляющий урок: несколько лишних секунд на выполнение обычной задачи делают эту задачу необычной. Ваша команда состоит не из лентяев, а просто из людей. Если выключатели поместить на уровне плеча, то все будут радостно гасить свет, уходя из комнаты, а если на уровне колена, то ваши счета за электричество будут астрономическими.

Как аукнется отсутствие проверки кода? Думаю, результаты будут серьезными. Посмотрите на коммиты Subversion за этот месяц: 55 из них продолжают какой-то другой коммит, а 35 имеют специальный маркер (см. *http://subversion.tigris.org/hacking.html#crediting*), указывающий, что это устранение проблем, найденных другими. По своему опыту работы с проектами могу сказать, что это обычный результат проверки коммитов. Таким образом, от 12 до 20% коммитов в Subversion являются результатом проверки предыдущих коммитов. Я не могу привести сравнительные данные по Emacs, потому что в Emacs нет таких стандартных условий атрибуции, как в Subversion. Но я уже давно слежу за разработкой в Emacs, и хотя некоторые коммиты там тоже являются результатами анализа предыдущих (а может, кто-то просто наткнулся на проблемный код в процессе совершенно другой работы), не верю, что этот показатель составляет от 12 до 20%.

Кроме того, выгоды своевременной проверки коммитов нельзя оценивать только по дальнейшим изменениям кода. Проверка коммитов поддерживает боевой дух, оттачивает мастерство (поскольку все учатся друг у друга), укрепляет способность к совместной работе (поскольку все привыкают к конструктивной критике со стороны других) и стимулирует участие (поскольку проверка публичная, и это подстегивает остальных). Лишить команду этих преимуществ из-за банального выбора пользовательского интерфейса – серьезная ошибка.

Они остаются в толпе: сказка о двух интерфейсах перевода

В 2005 году я написал книгу об управлении проектами с открытым исходным кодом[1], в основном посвященную ровно тому же, что и эта книга: внедрению инструментов для более эффективной работы команд.

Спустя некоторое время после того как книга была издана и опубликована в Сети, добровольцы стали переводить ее на другие языки. Я, естественно, очень обрадовался и, чтобы помочь им, сделал несколько инструментов для коллективной работы переводчиков. Обычно с каждым языком работал не один человек, и было бы разумно облегчить им совместную работу. К тому же определенные аспекты процесса перевода одинаковы для всех языков, так что даже разноязычным переводчикам было полезно видеть работу друг друга.

На бумаге все это выглядит прекрасно. Но случившееся в реальности оказалось горьким уроком и помогло понять, что бывает, когда неправильно определяешь задачи команды.

Что я им дал

Основу структурной базы, которую я предложил переводчикам, составляло расширение той, которую я сам использовал при написании книги, – подборка мастер-файлов XML, хранящихся в репозитории Subversion, написанных в DocBook DTD и конвертированных в формат вывода с помощью программ командной строки, управляемых Makefiles.

А теперь попробуем по-человечески:

Я написал книгу в формате (XML), который, хотя и неудобен для работы, но прост для создания веб-страниц и версий для печати. Вот так, например, в XML выглядит начало главы 3:

```
<chapter id="technical-infrastructure">

<title>Technical Infrastructure</title>

<simplesect>
<para>Free software projects rely on technologies that support the
selective capture and integration of information.  The more skilled
you are at using these technologies, and at persuading others to use
them, the more successful your project will be.  This only becomes
```

[1] «Producing Open Source Software» (O'Reilly; *http://producingoss.com/*) (Фогель К. «Создание программного обеспечения с открытым кодом», *http://www.producingoss.com/ru/*).

```
more true as the project grows.  Good information management is what
prevents open source projects from collapsing under the weight of
Brooks' Law<footnote>, <para>From his book <citetitle>The Mythical Man
Month</citetitle>, 1975.  See <ulink
url="http://en.wikipedia.org/wiki/The_Mythical_Man-Month"/> and <ulink
url="http://en.wikipedia.org/wiki/Brooks_Law"/>.</para></footnote>
which states that adding manpower to a late software project makes it
later.  Fred Brooks observed that the complexity of a project
increases as the <emphasis>square</emphasis> of the number of ...
```

А вот что из этого получается после конвертирования для веб-страницы:

Chapter 3: **Technical Infrastructure**

Free software projects rely on technologies that support the selective capture and integration of information. The more skilled you are at using these technologies, and at persuading others to use them, the more successful your project will be. This only becomes more true as the project grows. Good information management is what prevents open source projects from collapsing under the weight of Brooks' Law, which states that adding manpower to a late software project makes it later. Fred Brooks observed that the complexity of a project increases as the square of the number of...

(Глава 3. **Техническая структурная база**

Проекты свободного программного обеспечения основаны на технологиях, обеспечивающих выбор и интеграцию информации. Чем больше вы поднаторели в этих технологиях и в том, как убеждать других эти технологии использовать, тем более успешным будет ваш проект. И с ходом проекта эта истина только подтверждается. Хороший информационный менеджмент спасает проекты с открытым исходным кодом от краха под тяжестью закона Брукса, который утверждает, что если программный проект не укладывается в сроки, то добавление рабочей силы задержит его еще больше. Фред Брукс заметил, что сложность проекта возрастает как *квадрат* числа...**)**

Легко заметить схожесть обоих текстов, но читать мастер-файлы XML, безусловно, труднее. Но мне было удобно: недостатки работы в XML перевешивались способностью легко создавать разные версии вывода.

Свои XML-файлы я организовал с помощью системы управления версиями Subversion, хорошо знакомой программистам, но менее известной за пределами программистского сообщества. Конвертеры – один для Сети, другой для печатной версии – тоже были удобными и гибкими инструментами, но трудными в применении для непрограммистов. Только чтобы их установить, нужно было иметь на компьютере ряд поддерживающих их программ. Общий процесс в любом случае был довольно тяжелым, но если компьютер работал под UNIX-подобной операционной системой (что опять же общепринято среди

программистов, но необычно для других групп пользователей), было несколько легче, а если под Microsoft Windows – несколько труднее.

Все шло прекрасно, пока я был единственным автором. Я и сам программист, и был знаком с этими инструментами, а поскольку я уже тратил много сил на главное – написание книги, то немного поднажать и освоить несколько инструментов не представляло труда. Я знал, что это окупится с течением времени, так и вышло – для меня.

Но для переводчиков-добровольцев ситуация радикально отличалась. *Не все* из них были программистами, большинство до сих пор с подобными инструментами не работали, в итоге им многое пришлось осваивать с нуля – куда уж хуже! Далее, любой перевод выполнялся несколькими переводчиками, и для каждого из них причины разбираться с инструментами были более туманными, чем для меня: все тратили столько же времени, сколько я, но ведь ни один переводчик не писал целую книгу, и амортизировать затраты здесь не удавалось. Переводчики хотели проявить одну конкретную способность – взять английский текст и перевести его на другой язык; вместо этого им пришлось решать задачу, совершенно не связанную с этой способностью, – овладевать незнакомыми инструментами, которые в дальнейшем могли им и не пригодиться.

Удивительно не то, что некоторые взбунтовались, а то, что мы вообще получили сколько-нибудь удобоваримые переводы. На момент написания этой главы немецкий перевод вчерне готов, японский сделан на 80%, испанский – на 60% и так далее. (С французским интереснее, к нему мы еще вернемся.) Но то, через что пришлось пройти переводчикам, заставляет задуматься, насколько *больше* можно было бы сделать, если бы я не предъявил таких высоких программных требований к процессу. Взгляните на мои принципы перевода (*http://producingoss.com/translations.html#guidelines*) и поймите, чего я требовал от участников проекта. (Я бы привел их здесь, но они занимают слишком много места, что уже само по себе о чем-то говорит.)

Что я должен был им дать

К сожалению, я долго не понимал, насколько неудачна описанная выше система. Я знал, что она далека от совершенства, но трудно ведь сказать, почему данный конкретный переводчик делает значительно меньше, чем обещал, и уж тем более невозможно заметить, что переводчик отказывается работать, потому что напуган требованиями.

Но как-то я получил тревожную весть из Франции.

```
From: Bertrand Florat <...>
Subject: French translation
```

```
To: ProducingOSS Translators Mailing List
Date: Sun, 17 Feb 2008 16:54:32 +0100

Привет, Карл,

Просто хотели тебе сказать, что мы с Этьеном перешли на вики Framalang,
чтобы закончить книгу. Мы собираемся перевести ее всю в формат docbook сразу
по окончании работы (сейчас готово около 90%).

Думаю, было бы неплохо переместить французский перевод с http://www.
producingoss.com/fr/ на http://www.framalang.org/wiki/Producing_Open_Source_
Software, чтобы можно было пользоваться почти готовым переводом уже сейчас,
к тому же это может привлечь новых переводчиков.

Что скажешь?

Удачи,
Бертран
```

Иными словами, французские переводчики решили *обойти* техническую структурную базу, которую я задал, чтобы помочь в переводе. Вместо этого они скопировали все англоязычные данные в вики, где могли править документ в удобной и знакомой им среде. Затем[1] они собирались перенести свой перевод обратно в XML.

Суровый приговор моей структурной базе, надо сказать. Увидев сайт вики Framalang (*http://www.framalang.org/wiki/Producing_Open_Source_Software*), я не смог обвинять переводчиков. Он значительно более приспособлен для задачи координирования переводчиков, чем предложенная мной случайная система. В Framalang оригинальный текст и перевод отображаются рядом, разными цветами. Есть специальные функции для того, чтобы понять, кто и какие фрагменты переводил, для обратной связи, для создания централизованного списка согласованных общих терминов, для отметки того, что еще нужно сделать, и так далее. Все это можно было сделать и в моей системе, конечно, но дело в том, что там это получилось бы не сразу, пришлось бы снова и снова изобретать велосипед, а Framalang уже дает ключи от машины (рис. 21.6).

Заявление Бертрана подтолкнуло еще одного переводчика, и он тоже выразил недовольство доморощенной структурной базой.

```
From: "Manuel Barkhau" <mbarkhau@googlemail.com>
Subject: Re: French translation
To: bertrand@florat.net
Cc: producingoss-translators@red-bean.com
Date: Sun, 17 Feb 2008 17:21:35 +0100
```

[1] Собственно, перевод вчерне они закончили, когда я писал эту главу.

```
Reply-To: mbarkhau@gmail.com

Добрый день, друзья,

хотел бы спросить, не могли бы вы поделиться опытом и сравнить форматы вики
и Subversion. Участвует ли кто-то еще, кроме основных переводчиков. Сколько
переводчиков подключилось, качество их работы и т.д.
В текущем формате мне особенно недостает статистики читателей и обратной
связи.

Пока,
Мануэль
```

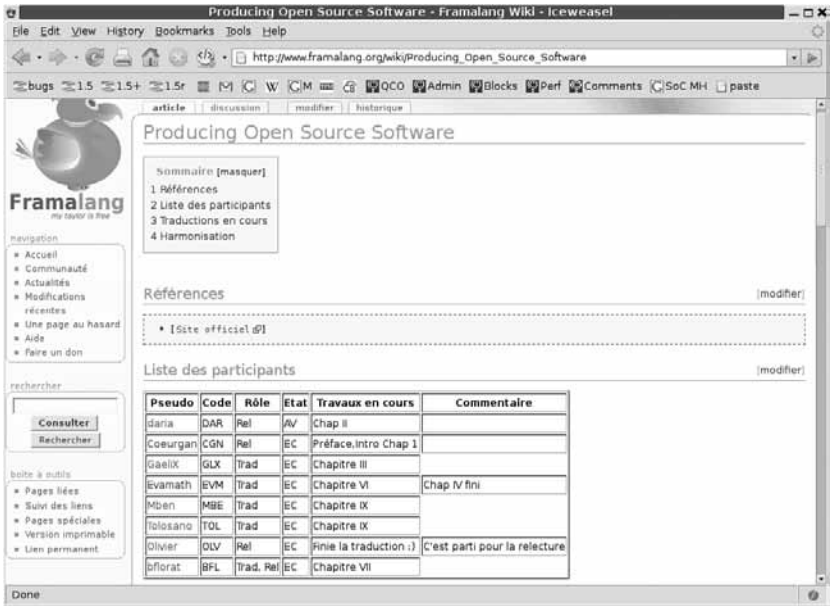


Рис. 21.6. *Перевод в вики Framalang*

Увидев эту переписку на листе рассылки, я припомнил некоторые разговоры с другими переводчиками. Многим из них было трудно работать в XML. Даже если сами переводчики вполне понимали формат, чтобы с ним работать, их инструменты редактирования были для этого не приспособлены. Их текстовые редакторы (особенно в Microsoft Windows) не всегда поддерживали XML. Сам я с этим никогда не сталкивался, но этого и следовало ожидать. Поскольку формат выбирал я, он, разумеется, хорошо работал с *моими* инструментами.

Сейчас мне уже очевидна связь между моими беседами с разочарованными переводчиками и ухудшением качества их работы – некоторые из них вообще вышли из проекта. Конечно, так бывает в любой группе энтузиастов, но тут важно заметить, что ухудшение качества связано с определенной причиной. Именно так все и было, но поскольку мне было удобно работать с моими инструментами, я долго не понимал, что они представляют существенную помеху для остальных. В итоге я эффективно погасил неизвестно сколько энергии добровольцев; многие из начатых переводов пока так и не закончены, и ответственность за это частично несу я, из-за своих нелепых инструментальных требований.

Заключение

Из предыдущих примеров можно вывести несколько принципов инструментов совместной работы:

- В хороших инструментах правильно понимается базовая единица информации.

  Например, в случае коммит-писем базовая единица – это изменение (или набор изменений). Если инструмент не рассматривает каждое изменение как единую логическую сущность, а вместо этого делит на несколько интерфейсных точек доступа (в данном случае электронных писем), то меньше вероятность, что изменения захотят проверять.

  Правильное понимание базовой единицы информации не просто облегчает применение данного конкретного инструмента. Оно облегчает совместную работу, давая членам команды говорить о том, над чем они работают, на одном языке.

- Если необходимые задачи постоянно откладываются («Постараюсь сделать это на выходных»), это часто знак того, что инструментарий требует от людей более высоких трудозатрат, чем они привыкли. Исправьте инструменты – и задержки могут исчезнуть.

  Contribulyzer не решает ни одну из интересных задач оценки работы участника: в конце концов, все равно приходится вручную перебирать изменения кода. Но он устранил *неинтересную* задачу – ручной поиск по логам управления изменениями с целью найти изменения конкретного участника. Такая перспектива настолько не вдохновляла (хотя эта работа подчас отнимала меньше времени, чем собственно оценка), что создавала помехи для работы, в основном просто потому, что была скучной.

  Устранив эту помеху и создав приятный и легкий интерфейс, мы перестали требовать от членов команды сделать над собой усилие, чтобы решиться на оценку участника. Теперь такое решение стало

напоминать заточку карандаша в электроточилке: чтобы начать работать, не требуется усилий, можно остановиться посередине и по желанию продолжить позже, можно заточить сколько угодно карандашей, не увеличивая общих усилий.

- Если ваша команда начинает обходить предложенные вами инструменты, прислушайтесь к ним: возможно, ваши инструменты плохи.

  Конечно, иногда бывает, что изначально выгоды применения новых инструментов просто недооцениваются, и тогда нужно просто привыкнуть. Но даже в этом случае отчасти может быть виноват сам инструмент, поскольку его преимущества изначально не ясны. По крайней мере, если все вокруг начинают искать обходные пути, то надо сесть и задуматься, что же ими движет. Когда французские переводчики моей книги перешли к вики-интерфейсу, не пожалев серьезных дополнительных усилий, это был явный признак того, что с моим интерфейсом что-то не так.

- Изменения в составе команды могут потребовать изменения инструментов.

  В случае с главами книги в XML, пока «команда» состояла из программистов, уже имевших предварительный опыт работы с файлами управления версиями XML (как было, пока команда состояла из меня одного), мои инструменты работали прекрасно. Но с приходом новых людей оказалось недостаточным просто распространить на них существующую систему. Вместо этого нужно было полностью изменить рабочую среду.

- Частичная автоматизация порой достаточна для улучшения работы.

  Тем, кто проектирует инструменты для совместной работы, нравятся стопроцентные решения: если что-то, по их мнению, *может* быть полностью автоматизировано, то так тому и быть. Но на практике усилия подобного рода не всегда окупаются; часто лучше прибегнуть к гибридной модели. Если можно получить 90% выгоды, затратив 10% усилий, то лучше так и поступить, помня, что команда всегда согласна пойти на изменения, если видит для себя в этом прямую выгоду. Например, Contribulyzer полностью зависит от людей, которые должны следовать определенным образцам при написании сообщений лога. Чтобы научиться этому, требуются время и силы. Но альтернатива – полностью автоматизированное интеллектуальное отслеживание проекта, которое устанавливает связи между сообщениями листа рассылки и коммитами, – требует огромных усилий, при этом гарантировать полную надежность нельзя.

  Собственно, это одно из тех преимуществ гибридных инструментов, о которых часто забывают: поскольку они заставляют людей оста-

ваться в контакте с данными, такие инструменты часто оказываются *более* надежными, чем якобы полностью автоматизированные. Цель хороших инструментов не в том, чтобы люди стали не нужны, но в том, чтобы они были удовлетворены. (Могут сказать, «чтобы они были эффективны», но ведь неэффективная команда редко удовлетворена собой, и к тому же вам вряд ли нужна эффективная, но недовольная команда.)

Несмотря на все выгоды применения хороших инструментов, опыт говорит мне, что в большинстве команд инструменты используются недостаточно по сравнению с их потенциальным применением. То есть в большинстве команд не пользуются потенциально огромными преимуществами. Думаю, дело тут в двух коренных предрассудках, разделяемых большинством.

Первый: опробовать новое слишком дорого. Полезность новшества редко заметна на вид, поэтому команды применяют новые инструменты выборочно, поскольку изменение привычек может пагубно сказаться как на производительности, так и на удовольствии от работы. (Не могу назвать этот предрассудок необоснованным. Его истоки очевидны, да я и сам его разделяю.)

Второй предрассудок тоньше. К хорошему инструменту быстро привыкаешь и уже не представляешь, как мог раньше без него обходиться, при этом забываются и обстоятельства *первого применения* инструмента. Пример этого феномена найти несложно: взять хотя бы тот текстовый редактор, в котором я все это пишу. 50 лет назад идея редактировать и форматировать текст одновременно с его набором была мечтой любого писателя;[1] а теперь это настолько вошло в привычку, что, пожалуй, в развитых странах даже школьники не представляют, как можно писать иначе. Но даже компьютерное редактирование текста кажется лишь добавочным улучшением по сравнению с таким подлинно преобразующим явлением, как всеобщая грамотность. Просто представьте, как бы вы организовывали работу команды, не опираясь на то, что все ее члены умеют писать и читать!

А теперь вообразите, как отреагирует ваша команда, если кто-то придет и скажет следующее:

> Я знаю, как многократно увеличить вашу производительность. Делайте, как я скажу, и ваша совместная работа улучшится сильнее, чем представлялось в самых радужных мечтах. Вот вам план: во-первых, все должны выучить этот набор символов – он называется *алфавит*. Эти символы называются *буквами*, и они примерно соответствуют звукам того языка, который для одних из вас родной, а для других нет. Когда вы

[1] *Улучшает* ли это сам текст – еще вопрос.

> освоите большинство букв, я покажу вам, как правильно группировать их, чтобы получались написанные слова. Выучив от 5000 до 10 000 слов, вы сможете передавать любые предложения, и тогда ваша производительность возрастет *до небес!*

Грамотность – классический пример хорошо окупающегося инструмента, который требует крупных долгосрочных вложений. Все тратят не много времени на овладение ею, но после этого она служит многие годы.

Обычно людей больше впечатляют инструменты, которые требуют меньших вложений и лучше окупаются. Но не стоит принимать в расчет только их, ведь и грамотность в конце концов окупается. Особая команда может выбрать не самый быстро окупаемый инструмент, особенно при разделяемой нагрузке: когда новым инструментом одновременно овладевает вся команда, это повышает чувство локтя. Инструмент окупается тем, что устраняет препятствия и открывает новые возможности. Мало что столь же непосредственно, эффективно и ежедневно формирует опыт совместной работы в техническом проекте, как применяемые командой инструменты. Чем глубже вы осознаете этот опыт, тем лучшие инструменты сможете выбирать.

22

Исследовательские команды

Интервью с Майклом Коллинзом

Область, в которой работает Майкл Коллинз, интересует нас по двум причинам. Во-первых, Майкл много занимался исследованиями в академическом учреждении. А во-вторых, его занятия были ориентированы на решение конкретных реальных проблем безопасности для серьезных потребителей в правительстве и вне его, так что его работа сочетает научный и коммерческий элементы. Нам захотелось, чтобы он рассказал об этом.

Эндрю: **Вы работали в исследовательском проекте по обнаружению вторжений в сети.**

Майкл: Главным образом мы занимались моделированием стандартов. В мои обязанности главным образом входило выявление аномалий, что является составной частью обнаружения вторжений. Выявление аномалий – это главным образом построение модели нормального поведения, чтобы, видя, что ваш объект вдруг отклонился от этой модели, вы сразу интересовались, в чем тут дело.

Приведу пример с кредитной картой, когда для вас сформирован стандарт расходов. И если вдруг вы начинаете тратить деньги в Катманду, то ваш банк звонит вам и спрашивает: «Вы в Катманду?» И ответ – нет. Это выявление аномалий в контексте кредитных карт. То же самое делается для сетевого трафика.

Эндрю: **Итак, вашей задачей было следить за данными маршрутизаторов, значит просто глядя на гигабайты ежедневных данных из логов маршрутизаторов вы можете обнаружить успешные и безуспешные попытки вторжения?**

Майкл: Ну, это уже Святой Грааль. Поначалу нужно было просто моделировать события. Но при этом вы даже не знаете, какие типы вопросов придется задавать. Это и есть краеугольный камень исследования: вы еще не знаете, что вам понадобится. Часто приходится создавать и переделывать инструменты прямо в процессе работы.

Сейчас клиент может сделать примерно один запрос за пять часов. А когда-то был один запрос за день. Тогда рисовали диаграмму и говорили: «Вот что мы нашли сегодня». Мы снизили время с 5-6 часов до 10 минут на двухпроцессорном Pentium (в 2001 году). И после первого отчета нас спросили: «Как вы смогли сделать столько запросов?» Мы объяснили, как уменьшили данные и формализовали процесс запроса. И в ответ от клиента услышали: «Мы это хотим».

Эндрю: **И внезапно у вас появился программный проект.**

Майкл: Вот именно.

Эндрю: **И вам понадобилась команда для работы над ним.**

Майкл: Да. Изначально нас было четверо: двое писали код, но никто из нас не считал себя «программистом». Например, когда я один раз назвал себя программистом, мой босс Суреш орал на меня минут пять с такими словами: «Можно работать с деревом, но не быть плотником. Ты исследователь. Ты пишешь код, чтобы получить ответы на вопросы, ты не разработчик».

Остановлюсь на паре моментов. Мы уже подготовили среду для тяжелой работы, и это еще мягко сказано. Например, имена наших файлов уже поддерживали систему управления версиями. Нам нужна была совместимость – как с предыдущими, так и с будущими версиями.

Мы достаточно поработали с техническими средами и знали, что прототипы – своего рода излишество, в том смысле что прототип от готового продукта порой отличается только названием. И не ожидали той реакции, которую получили, хотя с самого начала относились к разработке системы как к серьезной задаче.

Эндрю: **Работая над исследовательскими проектами и разговаривая с другими учеными, я понял, что ваш босс Сурeш прав: исследователи – не программисты. Представьте, что говорите с кем-то, кто только начал писать диссертацию или приступил к работе над проектом в университете; как они могли бы применить ваш опыт в собственных проектах? Где бы вы поступили иначе, чем более наивная исследовательская команда?**

Майкл: Думаю, во многом главное здесь то – и особенно это справедливо для выпускников, – что есть тенденция создавать множество программ из ничего, даже не думая о том, зачем они нужны.

Принципиальным моментом было решение раздробить работу на небольшие «проектики» с четкими целями. А одним из главных аргументов в пользу этого была необходимость убедиться, что код в «проектиках» жизнеспособен. Если взять архитектуру SiLK[1] (название системы, над которой мы работали), то в ней есть что-то вроде основной библиотеки, которая управляет чтением файлов, вводом/выводом и многим другим. И к ней написано около 40 приложений.

Исследование может легко обернуться неудачей. Поэтому нашей целью было оценить усилия по разработке этих небольших проектов, затем протестировать результаты, посмотреть, насколько они полезны, отвечают ли они на заданный вопрос, и если нет, то ликвидировать их. Мы планировали потратить определенное время, но таким образом мы, по крайней мере, не тратили слишком много времени. Самым важным оказалось то, что встроено в центральную библиотеку SiLK. Поскольку мы потратили много времени на управление версиями и проверку жизнеспособности центральной библиотеки, то, думаю, тем самым избавили себя от головной боли, присущей многим исследовательским проектам:

[1] SiLK (System for Internet-Level Knowledge – система интернет-уровня знаний) – коллекция инструментов для анализа трафика, над которыми работали Майкл и его команда. Она была выпущена как проект с открытым исходным кодом, скачать ее можно по адресу *http://tools.netsa.cert.org/silk/*.

в конце концов там остаешься с кучей программ, которая все растет. Думаю, дело здесь в том, что, занимаясь исследованиями, в определенном смысле торопишь процесс: хочется, чтобы идея появилась, была испробована и оказалась жизнеспособной. Мы почти безжалостно искореняли то, что не работало, и спокойно относились к неудачам. Много времени мы потратили на перестройку и поддержку центрального ядра системы в жизнеспособном и компактном состоянии.

Эндрю: **Как я понял, с точки зрения перспективы архитектуры, которую нужно постоянно иметь в виду, важно вовремя признать, что что-то не работает. И избавиться от неработающего элемента, чтобы не копить хлам, который все больше и больше мешает поддерживать код в удовлетворительном состоянии.**

Майкл: Да. Различие между исследованиями и насаждением возможностей может быть очень тонким.

Со временем у нас получалось так: допустим, инструмент X может что-то делать, а более поздний инструмент Y делает то же, но лучше. И тогда мы старались избавиться от инструмента X. Правда, оказалось, что некоторым было удобнее по-прежнему использовать программу X. Но разработка X становилась для нас менее приоритетной. Надо еще учесть, что поскольку использовалась действующая система, то с ней было связано множество документов: тренировочные курсы, руководство пользователя, собрания. В итоге фактически тренировочные курсы и руководства и определяли то, как использовалась система. Определив инструмент как неподходящий, мы удаляли его из основной части тренировочных курсов и списывали в архив.

Эндрю: **Итак, инструменты и архитектура — одна из областей, где вы попытались оптимизировать создание программ. Вы изменили свой подход к работе, чтобы облегчить поддержку. А как насчет совместной работы? Человеческого фактора? Не казалось ли вам, что нужно работать как-то иначе, чем в других исследовательских проектах? Было ли что-то, над чем пришлось существенно задуматься? Что заставляло тратить дополнительное время?**

Майкл: Что ж, это была интересная проблема, потому что мы столкнулись с тем, что некоторые профессионалы считают свои задачи приоритетными. У нас было несколько исследователей, в основном специалистов по статистике или высокоуровневому программированию, которые на самом деле не очень хорошо знают код. И были те, кого наняли именно как разработчиков. Но в идеале нам хотелось найти сотрудников, которые занимали бы промежуточное положение: умели проводить статистический анализ и писать код на C для численного анализа; на поиск таких людей и были направлены наши усилия. Во многом пото-

му, что наше направление работы нуждалось скорее в исследователях, чем собственно в программистах. Итак, нам нужен был такой полуавтономный программист-исследователь и к нему пара человек, которые были бы «хранителями» архитектуры. В долгосрочной перспективе, однако, это не сработало. Думаю, в итоге выпячивание своих задач неизбежно, просто потому что у всех есть свои специфические интересы и наборы навыков.

Одна из проблем, с которой мы столкнулись сразу же, заключалась в старшем исследователе, который был абсолютным *непрограммистом*. Так что если ему встречалась задача и рядом не было никого, кто написал бы ему код, он просто бил баклуши, и с этим нужно было что-то делать. В итоге нам пришлось пригласить дополнительных разработчиков, которые должны были работать с подобными людьми и писать им требуемый код.

Эндрю: **Итак, проблема состояла в том, что некоторые из членов вашей команды — чистые исследователи, ученые, математики, статистики — не являлись программистами. В программистской компании подобная команда обычно работает с бизнес-клиентами. Но в данном случае эти непрограммисты — важная часть команды.**

Майкл: Да. Мы столкнулись с проблемой требований в такой команде. Исследователи давали новые идеи. Это одна из причин, по которым нам нужны были одновременно исследователи и разработчики. Это должны были быть люди, которые брали задачу и создавали инструмент ее решения, жизнеспособный в различных ситуациях. Затем кто-то должен был понять, что сделано, и внедрить это в систему. То есть пара специалистов поддержки, которые встраивали бы в архитектуру прототипы, созданные программистами-исследователями. Один бы занимался их внедрением в библиотеки, и, думаю, это было бы хорошим решением: в том, что уже сделано, были некоторые пробелы, и указанный путь вполне позволял их заполнить. Если инструмент уже есть, можно расширить его функциональность дополнительными возможностями, это и есть целостный подход к задаче.

Преимущество состояло в том, что с возрастанием объема сделанного исследователи, не пишущие на C, могли бы заниматься сценариями или еще как-нибудь работать над внедрением инструментов. То есть у нас было решение — правда, медленное, — и можно было искать что-то получше. Таким образом, мы могли посвятить часть усилий разработке оптимизированной версии.

Эндрю: **То есть, по сути, это был столь же программистский проект, сколь и исследовательский, и члены команды, которые вообще-то не были программистами, могли приложить руку и к кодированию; это**

помогло бы свести концы с концами и перейти к следующему интересному научному вопросу.

Майкл: Верно. Идея состояла в том, чтобы найти золотую середину. Прежде всего, нельзя ведь ожидать, что чистый исследователь будет работать и над кодированием. Но если уже есть инструменты и исследователь способен написать сценарий командной оболочки для их использования, то это было бы и ему не зазорно, и нам небесполезно. Здесь и он мог бы продвинуться вперед, получив какой-то ответ на свой вопрос, и мы могли бы использовать информацию, чтобы сказать: «А теперь пора строить систему X».

И тут мы опять подошли мысли о том, что не знаем, чего будет стоить наш результат.

Эндрю: **Как часто вы ловили себя на том, что идете по ложному пути и что необходимо удалить код из программы? Ведь убирая функции или меняя код, порой вносишь столько же проблем, сколько решаешь.**

Майкл: Мы избежали этого следующим образом: у SiLK было две оболочки – архитектура, то есть файловая система, файлохранилище и тому подобное – и окружающие инструменты. С точки зрения SiLK исследование состояло из внедрения или сочетания набора инструментов, чтобы ответить на вопрос. Как правило, исследование касалось именно инструментов. Если мы выдавали какую-то идею и внедряли ее в качестве инструмента, то это никак не вредило центральной архитектуре.

Эндрю: **С точки зрения архитектурной перспективы, вы обеспечили высокую модульность: различные программы у вас скреплялись сценариями командной оболочки, что обеспечивало почти идеальную модульность. А с точки зрения команды вы убедились, что ваш персонал обладает максимальной технической гибкостью.**

Майкл: Да. В итоге мы остановились на тех, кто мог поддерживать архитектуру, а это техническая работа.

Эндрю: **Случались ли в вашей команде межличностные конфликты?**

Майкл: В группу исследователей входили решительные люди, каждый со своей точкой зрения и собственным подходом. Разумеется, обычным делом были споры.

Споры в основном можно было распределить по двум категориям: первая – исследовательские. Например: нужно попробовать идею X; нет, идею Y; нет, идею Z. Обычно идею действительно пробовали, и если она оказывалась верной, то развивали. А нет так нет. Но мы производили что-то и для клиентов. Все прототипы мы делали с помощью наших инструментов, и если эти инструменты оказывались полезными, то соз-

давали модуль и тренировочные курсы, которые учили применять этот инструмент. И я предложил учитывать требования тех, кто эти инструменты использовал, потому что обычно эти люди знали о них больше, чем мы сами: нам инструменты нужны были для научных целей, им – для практических!

Итак, это был первый тип споров. Другие споры велись, например, о целостности базы кода и тому подобном. У нас с Сурешем был потрясающий, исторический спор (с его стороны скорее дань тщеславию). Как-то на две недели его вызвали в присяжные. Он уходил, а мы как раз что-то сдавали, черт возьми. Обычно я отказывался сдавать продукт, пока сам его не протестирую. Дело в том, что тогда у нас еще не было специалиста по поддержке базы кода. И одной из моих обязанностей была надежность системы. Я прорабатывал все деревья ошибок, и по каждой ошибке я писал для системных администраторов, что делать с системой при таком-то сбое. И я не пропускал продукт. Наш спор длился четыре часа (я настоял на своем) и в итоге оказался важным для меня: до того я всегда был лишь подчиненным. Здесь же я понял, что это мой проект, что я не хочу, чтобы пятналось мое имя, поскольку он-то в это время был в суде присяжным.

Здесь надо отметить, что когда работаешь ученым, репутация всегда имеет огромное значение. Если пишешь работу, то ставишь на ней свое имя. И базовым принципом проекта было: если мы что-то выпускаем, то на кону наша репутация. Это понимали все члены группы. Ты представляешь нас, твоя работа представляет нас, если что-то идет не так, то это и твоя вина: нужно сразу все исправить и впредь быть осторожнее.

Это помогло нам, когда мы создали своего рода систему экспертной оценки продуктов. У нас были отдельные неудачи, и это я воспринимал как вызов личной чести. Однажды из-за неполадок в системе два поля поменялись местами, и я сказал программисту: «Послушай, из-за тебя я глупо выглядел перед клиентом. Ладно, забудем об этом, но больше так не делай». После он всегда был очень аккуратен и осторожен, и больше такого не случалось.

Итак, от теоретической платформы я перешел к мысли о том, каким должно быть техническое проектирование. Все соглашались с тем, что ни у кого нет монополии на истину. Так что споры предполагались. Также мы особо подчеркивали, что споры эти не содержат ничего личного. Я бы сказал, что большинство самых продуктивных команд состоят из людей, которые друг друга уважают, но находятся в нейтральных отношениях, так что могут высказывать непредвзятые суждения, не считая других дураками. Они не должны быть чрезмерно деликатными, чтобы не задеть чьи-то чувства, но и не должны стремиться сделать это нароч-

но. В группе все могут спорить со всеми, и так достигается объективность, а в конечном счете и консенсус.

Одно из наших правил спора заключалось в том, что в итоге всегда нужно достичь какого-то взаимного соглашения, а затем уже делать что угодно. По большей части мы спорили о технической стороне или экспериментах (и тому подобном), и после нужно было проверить, кто прав. Большинство тех, с кем мы общались, имели ученую степень. И проверка была неотъемлемой частью процесса, когда не знаешь, правильно что-то или нет.

23

Команда HADS

Карл Ремер

В древности человек, построивший арку, должен был стоять под ее сводом, когда снимали строительные леса. Если арка рушилась, то на голову нерадивого строителя. Когда я впервые попал на борт Боинга 777, я отчасти ощутил себя таким строителем: я написал некоторые критически важные элементы кода и работал в команде, разработавшей программные инструменты, с помощью которых была создана основная часть бортового ПО этого самолета. Это удивительное чувство – осознавать, что твоя жизнь зависит от качества написанной тобой программы. Как и при строительстве арки, конечный результат не был делом рук одного человека. Строитель, конечно, досконально знает свой проект, и порой эти знания не дают ему спокойно спать. Но зная всю подноготную разработки систем управления полетами и некоторые проблемы в ходе этой разработки, я мог чувствовать себя куда спокойнее, чем строитель арок. Самолет прошел множество проверок и испытаний, прежде чем его допустили к эксплуатации. Я не первый стоял под этим сводом. Очень волнительно ступать на борт самолета, зная, что твоя команда сыграла большую роль в разработке бортового ПО. Здесь я хочу рассказать историю команды HADS, которая разработала компилятор, систему поддержки выполнения, компоновщик, отладчик и другие программные средства поддержки, используемые разработчиками системы управления полетом на Боинге 777. Объем кода, написанный, внедренный и поддерживаемый этой маленькой командой, был примерно равен объему кода всех бортовых систем 777-го, над которыми работали сотни разработчиков. Мы не сорвали ни одного срока по проекту. Работа в той команде была одним из самых ярких этапов моей карьеры разработчика.

Полагаю, есть команды, изначально прекрасные и остающиеся такими в процессе работы. И есть ужасные команды, которые как были, так и остаются ужасными. Команда HADS вначале была еще далека от идеала, а в конце уже несколько поблекла. Но где-то в середине, на пике, это была идеальная команда, прекрасная во всех отношениях.

Предыстория

Теперь даже удивительно, что компания Honeywell Air Transport Systems (Авиатранспортные системы Honeywell) вообще оказалась задействованной в разработке комплекса инструментов HADS, поскольку обычно она занимается бортовой электроникой для коммерческой авиации.

История вопроса уходит в далекие 1980-е. Sperry Flight Systems переходила тогда от разработки аналоговых систем управления полетами к цифровым с применением компьютеров и программного обеспечения. На определенном этапе разработки в Sperry спроектировали собственный микропроцессор и дали задание команде разработчиков создать для него гибридный вариант Pascal. Они признавали, что проверка типов, предусмотренная в Pascal, давала некоторое преимущество по надежности и определенные возможности, важные при написании кода для систем управления полетами. Когда язык Ада стал стандартным, они заинтересовались им в плане разработки программного обеспечения для критичных с точки зрения безопасности систем управления полетами.

В 1985 году в Sperry Flight Systems задумались о применении Ада в электрической системе управления полетом, предлагаемой для Боинга 7J7. Целевой процессор должен был принадлежать семейству Intel 80*x*86 со средой разработки Ада от поставщика компиляторов DDC-I. DDC-I был признанным поставщиком компиляторов и славился, в частности, качеством своих программных инструментов и среды выполнения для процессоров семейства Intel 80*x*86. Удобно было и то, что у DDC-I были торговое представительство и конструкторский отдел в Финиксе (Аризона), где находилась и Sperry Flight Systems.

Процесс разработки ПО, критически важного для обеспечения полетов, обязательно предполагает документирование всех процедур, входящих в систему, а также написание тестов, обеспечивающих покрытие кода. Началась работа по созданию документации и тестов для среды выполнения DDC-I на Ада на уровне, требуемом согласно DO-178B – стандарту, принятому Федеральным управлением гражданской авиации США для разработки ПО бортовых авиационных систем. Где-то в разгаре этого процесса Sperry Flight Systems была продана компании Honeywell.

Подразделение, занимавшееся крупноразмерными воздушными судами, такими как реактивные самолеты коммерческой авиации, получило название Honeywell Air Transport Systems Division (Подразделение авиатранспортных систем Honeywell).

В то время мы с женой были старшими преподавателями в университете Пердью (Форт Уэйн, Индиана). Летом мы оказывали консалтинговые услуги по ПО подразделению Magnavox в Форт Уэйне. Magnavox тогда одной из первых перешла на Ада, запустив свой первый по-настоящему крупный проект на Ада – AFATDS (Advanced Field Artillery Tactical Data System – усовершенствованная боевая информационно-управляющая система полевой артиллерии). В 1988 году мы решили, что жена продолжит свою академическую карьеру, готовясь к защите кандидатской степени в области информационных технологий, а я буду обеспечивать семью, найдя работу по специальности.

Мой опыт работы с Ада удовлетворил требованиям Honeywell к кандидатам, и меня приняли в небольшую команду, которая занималась обратной разработкой, документированием и написанием тестов для покрытия кода среды выполнения DDC-I на Ада, которая должна была применяться в проекте 7J7. В целях ускорения работы система была написана на ассемблере, поэтому первым этапом ее документирования была обратная разработка и создание обобщенного описательного псевдокода для каждого алгоритма. Документировались также назначение каждой подпрограммы и момент ее вызова. Команда уже практически завершила создание документации для среды выполнения, когда Боинг закрыл проект 7J7.

Авиатранспортные системы Honeywell не утратили интерес к Ада и были уверены, что в будущем появятся новые проекты в этой области, поэтому я приступил к разработке стандартов проектирования и программирования на Ада для Honeywell. Я организовал также внутренние обучающие курсы по разработке на Ада. Обе эти задачи были важны, поскольку в Honeywell было мало разработчиков, имевших опыт работы с Ада или опыт использования многочисленных принципов разработки ПО, реализуемых на основе Ада. Можно сказать, я был домашним экспертом компании по языку Ада.

Вскоре Боинг приступил к разработке нового воздушного судна – Боинга 777. Авиатранспортные системы Honeywell получили крупный заказ на разработку ПО.

Среди программ, которые предстояло разработать, была система AIMS (Автоматизированная система управления техническим обслуживанием и ремонтом авиационной техники). Эта система должна была быть целиком написана на Ада. Если в предыдущем бортовом ПО для каждого конструктивно-сменного узла, КСУ (Line Replaceable Unit,

LRU)) нужен был отдельный процессор, AIMS позволяла объединить в единую систему функциональные возможности нескольких КСУ. Один процессор мог запускать несколько приложений. В критичных для полетов системах необходимо, чтобы базовое ПО обеспечивало следующее: одно приложение не может отнимать время другого процесса или искажать его данные. Поскольку на одном процессоре должны были исполняться несколько критичных ко времени приложений, которые использовали единую информацию, системе AIMS требовался очень мощный процессор.

Вес является важным параметром для воздушных судов. Каждый килограмм требует дополнительного горючего. Для экономии веса система AIMS проектировалась как пассивно охлаждаемая – для охлаждения не предусматривался вентилятор.

Специалисты Honeywell по аппаратному обеспечению провели тщательный анализ имевшихся на рынке процессоров и отбраковали самые популярные платы Motorola и Intel, поскольку те либо потребляли слишком много энергии (и, следовательно, не могли охлаждаться пассивно) либо не обладали достаточной мощностью для выполнения этой задачи. Проведенный анализ и ранние прототипы, разработанные инженерами по аппаратным средствам, привели их к выводу, что для системы AIMS на Боинге 777 необходим процессор AMD 29050. Базовой системой для разработки была выбрана VMS на компьютерах DEC VAX, а затем на рабочих станциях DEC VAX.

Тут возникла интересная проблема: Honeywell обещала компании Boeing разработать систему AIMS на языке Ада, но для процессора AMD 29050 не было компилятора Ада. Как главный специалист по Ада я рекомендовал пересмотреть выбор процессора AMD 29050. Поскольку качество компилятора и инструментов поддержки имело огромное значение для разработки, я считал, что выбор полноценного компилятора имеет ключевое значение. Специалисты по аппаратному обеспечению настаивали, что только процессор AMD 29050 удовлетворяет их потребностям, и потому Honeywell должна подыскать систему разработки. Они запросили несколько поставщиков компиляторов Ада о возможности создания необходимого компилятора и сопутствующих инструментов, но по разным причинам все они отказались. Тогда Honeywell решила разработать собственные компилятор и инструменты поддержки. Ведь прежде она уже разрабатывала компиляторы. При этом мы понимали, что разработать компилятор и инструменты для Ада сложнее, чем для Pascal, поэтому и пригласили DDC-I к сотрудничеству. В качестве отправной точки предполагалось использовать передовые технологии DDC-I. Кроме того, два сотрудника DDC-I должны были консультировать проект. В проекте также должны были работать несколько инже-

неров из отдела ПО компании Honeywell. Первоначально я в эту команду не входил, продолжая выполнять роль эксперта. Разрабатываемый продукт назывался HADS (Honeywell Ada Development System – Система программирования на языке Ада компании Honeywell).

DDC-I только что завершила проект разработки системы программирования на Ада для процессора Intel I960. Поскольку, как и AMD 29050, это RISC-подобный процессор, в качестве базы для запуска нашего проекта был выбран исходный код этого проекта. В компиляторе Ада задействован не только компилятор, генерирующий исходный код. Требуется также среда выполнения для управления задачами, обработки исключений и управления памятью. Управление задачами в Ада – это способ в рамках языка создать возможность для коммуникации и синхронизации независимых потоков управления. Управление исключениями, которое сейчас широко применяется в языках C++ и Java, тогда применялось редко. Управление памятью в Ада позволяет управлять как общей кучей, так и отдельными специализированными кучами.

Компилятор и компоновщик относятся к программным средствам хост-системы; в DDC-I использовали родной для DEC VAX компилятор Ада в качестве средства разработки компилятора Ада для I960, поэтому и для нас она стал естественным выбором при написании компилятора и компоновщика для HADS.

Среда выполнения компилятора представляет собой код, который запускается в целевой системе. Поскольку у нас еще не было компилятора в HADS, который генерировал бы необходимый код, среда выполнения была написана на C и ассемблере в AM29050. Те исполняемые элементы, которые войдут в критичные по безопасности приложения окончательной версии AIMS необходимо будет задокументировать и протестировать в соответствии со стандартами DO-178B.

Поскольку я присоединился к команде позже, мои воспоминания о начальном этапе ее работы представляют собой то, что я слышал краем уха в офисе, или то, что мне рассказали. Некоторые имена вымышлены, отчасти потому, что я забыл настоящие, отчасти во избежание каких-либо осложнений.

Первый состав команды

Поскольку мой отсек находился там же, где рабочие места членов команды HADS, я мог наблюдать их работу в начале проекта. И начало это не предвещало ничего хорошего.

Том и Дэйв работали на проекте как консультанты от DDC-I; Том руководил проектом 1960 в DDC-I. Дэйв, помимо прочего, играл ключе-

вую роль в решении одной из самых сложных задач процесса создания кода – эффективного применения регистров в аппаратном обеспечении. Ни у кого из инженеров Honeywell не было опыта разработки компиляторов, разве что в рамках университетской программы. Том не сразу стал руководителем проекта HADS, но с самого начала оказывал большую консультационную поддержку.

Кроме определения примерного графика работы в первую очередь нужно было проанализировать разницу между Intel I960 и AMD 29050 и понять, какие элементы того проекта можно сохранить, какие требуют небольшой доработки и какие нужно полностью переписать. Это должны были сделать инженеры Honeywell, а Том и Дэйв – консультировать их. Крису предстояло проанализировать различия в архитектуре, которые могли сказаться на компоновке. Джон анализировал влияние архитектуры на управление исключениями. Синтия занималась компоновкой, Уэйлин – управлением памятью. Аджит работал над постановкой задач в Ада, а также решал вопросы, связанные с тем, что называется User Configurable Code (код, формируемый пользователем), поскольку они тесно связаны с управлением задачами в Ада в части планирования. Выполнив анализ, инженеры должны были затем проектировать эти средства разработки. В процессе внедрения предполагалось задействовать дополнительные инженерные ресурсы Honeywell включая Дана, Фреда, Уиллиса и Генри.

Очень интересно наблюдать за тем, насколько по-разному люди работают. Из моего отсека было хорошо видно, что происходит в проекте. Джон обходился без чьей-либо помощи, большей частью работая над документами самостоятельно. Уэйлин обычно не просила помочь Тома и Дэйва. Поскольку прежде мы вместе работали над документацией проекта 7J7, она часто обращалась с каким-нибудь вопросом ко мне. Крис проводил много времени у Тома, но большей частью плакался, как непосильна его задача. Том не очень-то сочувствовал Крису и не делал работу за него, но старался указать ему на то, над чем стоит подумать, или на документы, откуда можно почерпнуть информацию. Фред был из тех, у кого всегда хватает ярких идей, но недостаточно терпения и упорства, чтобы их реализовать. Помнится, я не раз ловил себя на мысли о том, как мне повезло, что я не в их команде.

Спустя несколько недель Джон решил, что он не справляется, и попросил перевести его на другой проект. Поскольку у меня был опыт документирования среды выполнения Ада, мне предложили занять его место. Несмотря на плохие предчувствия, отказ был бы не самым мудрым политическим решением, и я согласился работать в этой команде, приняв на себя ответственность за управление исключениями. Совещание по обсуждению различий было не за горами, но оказалось,

что переданная мне Джоном документация была в весьма неплохом состоянии. К нашему первому обсуждению проектирования в общем виде я уже был готов отчитаться по основным вопросам архитектуры, которые могли повлиять на управление исключениями. Но поскольку я все же вступил в игру позже, эта часть проекта была в зоне риска.

К той же встрече Крис подготовил презентацию по различиям в архитектуре, которые могли повлиять на генерацию кода. Каждому на той встрече было ясно, что он практически ничего не сделал. Он начал с одного из документов DDC-I по процессору I960, в который внес несколько мелких поправок. Многие разделы его отчета описывали только I960, даже не упоминая AMD 29050. Иногда казалось, что единственное исправление состоит в автозамене «I960» на «AMD 29050»; кроме того, некоторые выдвинутые им положения были явно ошибочными. Все это было очень плохо, ведь генерация кода – один из самых важных элементов компилятора. Полагаю, я был слишком поражен этой презентацией, чтобы запомнить реакцию других участников совещания. Не помню, чтобы Криса критиковали напрямую, но, разумеется, ему указали на определенные ошибки и упущения в презентации. Та встреча вовсе не укрепила мою уверенность в том, что проект увенчается успехом.

Позже Дэйв рассказал мне, что возвращаясь на рабочие места, они с Томом обсудили, как мало сделал Крис. Они даже представить не могли, что результат нулевой. И решили обратиться к главе группы из подразделения программных средств Honeywell с просьбой убрать Криса из нашего проекта. Интересно, что одновременно сам Крис сразу после того совещания попросил руководство перевести его на другой проект. Некоторые другие области анализа, хоть и не были в столь плачевном состоянии, оставляли желать лучшего. Таким образом, первый контрольный срок, к которому планировалось скомпилировать, скомпоновать и запустить программу «Hello World»[1], был под угрозой. Многие в команде не справлялись с нагрузкой.

Исправление ситуации

Необходимо было что-то предпринять, чтобы исправить ситуацию с проектом. Команда была реорганизована и, несмотря на мрачно мая-

[1] Автор иронично обыгрывает название программы, результатом работы которой является вывод указанной фразы. Обычно такая программа является первым опытом при изучении нового языка и используется в учебных целях для определения ее базовой структуры. Традиция использования фразы «Hello, world!» в качестве тестового сообщения берет начало в 1978 году. – *Прим. ред.*

чившие впереди дедлайны, уменьшена. Криса так и не заменили. Фред ушел в другой проект. Уиллис и Генри занялись другой работой по проекту Боинга 777 – компилятором C, на котором был написан небольшой фрагмент кода для 777-го. Пришел Питер – дополнительный консультант из DDC-I. Том официально возглавил проект. Команда поделила между собой несделанную работу – определение уникальных характеристик процессора, которые влияли на генерацию кода и реализацию среды выполнения системы. Внешний интерфейс компилятора можно было не менять. Чтобы скомпилировать и запустить программу Hello World, нужен был небольшой объем сгенерированного кода и минимальная среда выполнения. Этот первый этап мы сдали практически вовремя.

После этого некоторые участники команды приступили к разработке рабочего проекта и реализации управления памятью, задачами и исключениями в среде выполнения системы. Другие продолжили работать над генерацией кода.

Уэйлин разрабатывала и внедряла функции управления памятью. В общем виде многие алгоритмы могли повторять те, что применялись для процессора I960. Главная сложность заключалась в том, что они были написаны на языке ассемблера, поэтому требовалось полностью переписать код. И Уэйлин пришлось освоить язык ассемблера.

Низкоуровневая часть кода, формируемого пользователем (User Configurable Code, UCC), позволяет написать очень небольшой фрагмент кода, обеспечивающий взаимодействие среды выполнения с конкретной аппаратурой. Примеры UCC-кода: низкоуровневый код для ввода/вывода, код для управления таймерами и код для обработки прерываний. На данном этапе были разработаны и внедрены эти элементы. UCC-кодом занимался Аджит.

Большинство алгоритмов управления задачами можно было оставить теми же, что и для процессора I960. Главные сложности заключались не в конструкциях языка Ада для управления задачами, а в общей поддержке независимых потоков управления, то есть в том, как организовать переключение контекста и управление прерываниями. Поскольку управление прерываниями и переключение контекста очень близки к аппаратному управлению, выполняемому при помощи UCC-кода, над этой частью кода также работал Аджит.

Я занимался вопросом проектирования и реализации управления исключениями. Алгоритмы подходили по общему виду и назначению, но на нижнем уровне реализации архитектура сильно отличалась. Эта часть кода была написана на C. Для AMD 29050 имелся компилятор C компании Metaware, но не было отладчика исходного кода. Симуля-

тор машинных команд позволял запускать программы и отлаживать машинный код. Компилятор мог генерировать листинги – смесь исходного кода и ассемблера, позволяющие привязать исходный код к сгенерированному машинному коду при отладке. Отсутствие отладчика исходного кода поначалу сильно мешало, но мы ухитрились создать приемлемую среду разработки с помощью листингов и отладчика машинного кода. Плюс мне пришлось вникнуть в машинный язык. Эти знания очень пригодились мне позже, в разработке элементов генератора кода.

Синтия продолжала работать над компоновкой и дополнительными структурами в тех областях, где генерировали код другие участники команды.

В рамках стандарта для Ада (Ada 83) был определен набор тестов под названием Ada Compiler Validation Capability, ACVC (возможность проверки компилятора Ада). Чтобы проверить корректность работы, компилятор и соответствующие элементы среды выполнения должны пройти все применимые тесты ACVC для его окружения. После сроков первого этапа – возможности запуска программы Hello World – в качестве следующих этапов было определено прохождение определенного процента тестов ACVC.

К одному из дедлайнов надо было пройти 80% тестов ACVC без управления задачами. После этого мы сделали первый релиз системы для пользователей из команды проекта 777. Многие группы пользователей еще не могли начать программировать на этом этапе, но были и те, кто готов был приступить к созданию прототипов ПО. Одна из групп, которой руководил Джоэл, с особым интересом ждала наших результатов, поскольку он слыхал о несовершенстве языка Ада. И хотя многие из этих так называемых недостатков устранялись отключением после первого этапа разработки проверок на этапе выполнения, определяемых языком и заданных по умолчанию, группа была настороже. Как только вышел первый релиз компилятора, они проанализировали получаемый машинный код и пришли к выводу, что компилятор недостаточно хорош. И хотя мы пытались донести до них, что акцент в этой первой версии делался не на функциональных характеристиках и что последующие релизы будут гораздо лучше оптимизированы, они начали искать альтернативу. Практически безуспешно, поскольку у Honeywell были обязательства по использованию языка Ада, а других компиляторов не было. Они решили, что большую часть своего кода реализуют путем вставки машинного кода с помощью специального механизма Ада. При этом пользователь пишет свой код непосредственно на языке ассемблера, а компилятор просто подключает его. То есть

технически выполняется обязательство программирования на Ада, но с нужной функциональностью. После выпуска примерно четвертой версии компилятора команда HADS еще раз проверила оригинальный код, созданный той группой, и выяснилось, что компилятор генерирует гораздо более эффективный и менее громоздкий код, чем в «сборном» варианте. Это совсем не удивительно и подтверждается целым рядом исследовательских публикаций. Компилятор языка высокого уровня – в частности, компилятор Ада – может отслеживать и использовать намного больше информации, чем помещается в голове программиста. Кроме того, разработчики компилятора обычно лучше осведомлены об особенностях архитектуры, которые можно задействовать. Команде HADS приходилось постоянно «продвигать» свой продукт.

Проекту очень повезло с процессором AMD 29050 в одном: имелся очень хороший и полный симулятор. Это означало, что разработчики могут создавать и тестировать код, в то время как система, для которой он предназначался, еще находилась на стадии раннего прототипа. Оказалось, что любой код, предназначенный для целевой системы, успешно выполнялся и на симуляторе. Только в нескольких случаях, связанных с синхронизацией, недопустимый для целевой системы код нормально работал на симуляторе. Наличие такого симулятора стало важным фактором, позволившим нашей команде держаться графика работ по проекту.

И хотя иногда мы бывали на грани срыва сроков, команда HADS вовремя выполнила все этапы по созданию компилятора, который прошел тесты ACVC на 100%. Когда эта цель была достигнута, до полного завершения работы было еще далеко. Например, Ада определяет и допускает целый ряд возможностей, зависящих от реализации. К ним относится, например, возможность точно описать расположение структур данных, таких как строки и массивы. Нужно было реализовать характеристики, приемлемые для процессора AMD 29050 и для проекта Боинг 777. Кроме того, требовалось добавить обещанную оптимизацию. Когда были реализованы фрагменты UCC-кода по нашему проекту и управление задачами на Ада, Аджит ушел из нашей команды в другую группу проекта 777.

Работа с пользователями

Одной из наших ключевых задач была организация обучения и поддержки пользователей данного комплекса инструментов. Большинство из них не были знакомы с языком программирования Ада, очень немногие разбирались в архитектуре процессора, да и сам комплекс инструментов был новым для всех пользователей.

Кроме развенчания некоторых мифов об Ada, команде HADS приходилось бороться с общим недостатком знаний об этом языке среди разработчиков проекта 777. Ада – очень мощный язык с множеством функциональных возможностей, делающих его идеальным для встроенных систем, критически важных с точки зрения безопасности. Сегодня он уже не считается сложным для изучения языком, как было тогда. Многие концепции разработки ПО, в поддержку которых создавался Ада, были незнакомы большинству разработчиков. Команде HADS пришлось взять на себя неофициальную роль инструктора по Ада. Например, строгая проверка типов в Ада может затруднить компиляцию программ. Это конструктивная особенность языка, позволяющая выявлять несоответствия как можно раньше (в процессе компиляции), а не во время выполнения. Но пользователи обычно винят во всех трудностях инструмент. Нас часто вызывали помочь какому-нибудь пользователю решить проблему при компиляции. Поскольку большинство членов команды HADS были хорошо знакомы с Ада, нам несложно было помогать, но это отнимало много времени. И могло порождать известную напряженнность во взаимоотношениях.

Помню, однажды меня попросили помочь Фрэнку, у которого были сложности с компиляцией кода. Я подошел к нему, и мы вместе просмотрели его код. Выяснилось, что у него тип элементов массива не соответствовал описанному типу массива. Мы исправили это. Фрэнк был очень недоволен такой строгой проверкой типов. Думаю, он был из тех программистов, кто считает, что все должно компилироваться, а отлаживать нужно результат. В конце концов мы довели его программу до того, что ее стало можно откомпилировать, но на вид она сильно усложнилась. Я сказал: «Думаю, можно сделать это по-другому», и познакомил его с некоторыми функциями Ада, которые могли сделать программу яснее, а возможно и сгенерировать более эффективный код. Фрэнк очень огорчился, восприняв критику его кода как оскорбление. Я не ожидал этого, поскольку в проекте HADS мы всегда так поступали, и вся команда считала это полезной практикой. Конечно, члены команды HADS знали, что они хорошие специалисты, и не чувствовали себя под угрозой. А Фрэнк немало помучился, чтобы хоть что-то скомпилировать, и теперь ему говорят, что можно было сделать это более эффективно.

Прохождение полного комплекса тестов не гарантирует того, что в компиляторе нет дефектов. Тесты ACVC представляют собой набор очень жестких проверок базовых функций Ада, но комбинации некоторых характеристик могут остаться непроверенными, а конкретную реализацию вообще невозможно проверить. Безусловно, у команды HADS бывали ошибки в релизах. Очень удачной находкой было то, что мы

сопровождали каждую ошибку небольшим тестом, который добавляли в наш комплекс регрессионных тестов, запускаемый в дополнение к тестам ACVC. Создать такой набор тестовых данных с нуля могло бы быть непросто.

Однажды Пол показал команде отчет о критической ошибке, и когда мы его рассмотрели, то пришли к выводу, что, скорее всего, проблема в генерации кода. К сожалению, по словам Пола, ошибка появлялась только в больших, сложных системах, а когда он пытался упростить их, ошибка исчезала. Очевидно, он обнаружил некую проблему превышения определенных допустимых параметров. К счастью, поскольку все мы работали в Honeywell, нам легко было получить права доступа ко всему коду, разработанному Полом. Включив вывод некоторых внутренних параметров компилятора, мы смогли увидеть, что именно делает компилятор, сократив первоначальный набор данных из нескольких тысяч строк кода до тестовой программы примерно в двадцать строк. Не всегда проблемы решались так легко, порой на небольшой контрольный пример уходило несколько дней. Однако эти усилия были оправданны, поскольку найти возможность исправления критической ошибки в небольшой программе гораздо проще, чем в большой. Также на основе маленькой программы мы составили регрессионный тест.

Я часто говорил, что программные инструменты находятся в самом конце пищевой цепочки. То есть что когда что-то отказывает, пользователь в первую очередь обвиняет в этом инструмент. Мы постоянно с этим сталкивались, поскольку в нашей ситуации не только был инструмент, на который можно было свалить все проблемы, но он еще и был «доморощенным», то есть доверяли ему куда меньше. Однажды мне позвонила Фрэнсис с жалобой на ошибку в компиляторе. Когда она писала в массив, из-за ошибки генерации кода переписывалась еще и соседняя переменная. Такое поведение возможно и даже вероятно при использовании языка C, в котором не проверяется, чтобы массив не занимал больше места, чем ему полагается по описанию. Но в Ада проверки при компиляции и на этапе выполнения должны выявлять такое поведение. Если статистически определено, что массив может превысить допустимый размер, во время компиляции будет выдано предупреждение. На этапе выполнения проверка вызовет исключение, если будет предпринята попытка доступа к массиву по индексу, который больше допустимого. Такие проверки на этапе выполнения генерируют дополнительный код и могут создавать проблемы, поэтому в компиляторах Ада есть возможность отключить некоторые или все проверки на этапе выполнения. Фрэнсис делала сборку по старому сценарию сборки, в котором были отключены проверки на этапе выполнения. Небольшое расследование и отладка показали, что значения попадали за пределы

массива в исходном коде. Компилятор только генерировал код, строго выполняющий заданные инструкции. Нам постоянно приходилось рассказывать пользователям о параметрах командной строки инструментов и об их возможностях. Мы часто видели отчеты о критических ошибках, которые оказывались в итоге ошибками пользователей. Между собой мы называли их ошибками ТП (Тупого Пользователя).

Пройдя тесты ACVC и выпустив несколько версий, где были внедрены и некоторые возможности, зависящие от реализации, команда начала работать над созданием среды отладки системы HADS на уровне исходного кода. Примерно к тому времени Синтию перевели из команды в проект 777. Оставшиеся члены команды выполнили первый этап работы по отладчику. Затем Том, Дэйв, Питер, Дэн и Уэйлин вернулись к компилятору, а я завершал первую версию отладчика. Том вел локальную оптимизацию, Дэйв работал над эффективным использованием регистров, а Дэн и Уэйлин занялись общей проблемой выбора оптимальной генерации машинного кода.

Для эффективного использования регистров Дэйву, в частности, нужно было проводить анализ «действующий/не действующий» (live/dead analysis), чтобы определить, хранит ли регистр переменную или она уже не нужна. «Побочный эффект» этого анализа состоял в том, что было легко определить, считывается ли регистр до того, как в него что-то записано. То есть что используется неинициализированная переменная. Дэйв сгенерировал код, выводящий предупреждение об использовании неинициализированной переменной. Когда вышла эта версия, пользователи начали получать такие предупреждения и стали звонить нам, прося помочь найти неинициализированные переменные. Дэйв нашел остроумный способ выводить имя неинициализированной переменной, анализируя ввод по адресу регистра. Он внедрил эту функцию и написал несколько небольших контрольных примеров, которые показывали, что она работает как надо. Мы включили эту функцию в следующий релиз. И почти сразу же мы начали получать жалобы на то, что компилятор стал работать медленнее. Разобравшись, мы обнаружили, что все дело в этом анализе неинициализированных переменных. Для одной из систем, которая в прежней версии компилировалась пять минут, нам пришлось прервать компиляцию: она шла уже второй день. И хотя в отсутствие неинициализированных переменных работа компилятора совсем не замедлялась, было очевидно, что возможность определить имя любой найденной неинициализированной переменной обходилась слишком дорого.

Добавление информации о неинициализированных переменных – только один из примеров запроса на функцию, поступившего от пользователей. Поскольку мы находились в той же компании и были знакомы уже

со многими пользователями, часто к нам просто подходили и просили добавить ту или иную функцию в какой-нибудь инструмент. Этот процесс грозил стать неуправляемым, и мы решили «просто отказывать» в любой такой просьбе. Всех просителей мы направляли к нашему руководителю проекта, а он уже решал вместе с командой, включим ли мы данный запрос в работу и какой приоритет ему назначим.

А еще мы для каждого завели «топ 10» – список из 10 вопросов с убывающими приоритетами. И каждый знал, над чем он должен работать сейчас и что последует за этим. Том мог пересматривать такие списки ежедневно, но если мы не соглашались с приоритетами, это можно было обсудить с ним. Кроме того, по пятницам днем у нас были собрания, где мы обсуждали успехи прошедшей недели и вместе определяли приоритетные направления на следующую. Иногда на этих собраниях обсуждались и технические вопросы.

Разрабатывая отладчик исходного кода, мы во многом могли опираться на предшествующую работу DDC-I, но отладка обычно привязана к архитектуре. Обработка отладочной информации зависит от архитектуры и по каждой целевой системе есть свои способы работы, например задание точек останова. Поскольку симулятор играл очень важную роль в работе команды, был разработан отладчик, который работал как с симулятором, так и с целевыми системами. Как и с компилятором, выпуск отладчика не был подобен «большому взрыву» – скорее, планировалась серия релизов с добавлением тех или иных функций.

Для выпуска отладчика потребовалась еще большая помощь пользователей. Один из очень запутанных вопросов представляли переменные в регистрах. Код, перенесенный из базового проекта DDC-I, нам очень помог, поскольку в нем было указано начальное размещение переменной в регистре, а также то, где сохранялся регистр, если вызываемая подпрограмма относилась к области видимости переменной. Но приступив к оптимизации с целью повторного использования регистров, мы обнаружили, что не можем отследить, как переменная перемещается в память, а занимаемый ею регистр выделяется для другой переменной. Пользователь считывал значение переменной из регистра, в котором этого значения уже не было, поскольку регистр уже использовался для других целей. В результате он получал неправильное, сбивающее с толку значение, тогда как верным был бы ответ, что в данном месте кода эта переменная больше недоступна. Серьезной задачей для нас стала разработка системы, позволяющей отслеживать это, и ее внедрение в отладчик и те части компилятора, которые отвечали за назначение регистров.

Многие пользователи были незнакомы с отладчиками исходного кода и получали информацию о своих программах в основном с помощью операторов печати. Мы постоянно обучали их пользоваться отладчиком. Бывало ощущение, что руководство пользователя никто и никогда не читает. Едва знакомившись с отладчиком, пользователи часто просили добавить какую-либо функцию для упрощения процесса отладки. Удовлетворить подобную просьбу обычно было проще простого: эта функция уже имелась в отладчике, пользователь просто не читал руководства.

Даже при отладке исходного кода пользователям иногда полезна возможность отладки машинного кода, чтобы проанализировать поведение кода на нижнем уровне. Разумеется, для отладки на уровне машинного кода полезно что-то знать об архитектуре машины и ее языке ассемблера. К сожалению, многие пользователи не обременяют себя чтением такой документации. Однажды мне позвонил Сэм: «Тут то ли отладчик глючит, то ли что-то с симулятором!» Я прошел к нему в отсек, чтобы попытаться определить и, по возможности, устранить проблему. Сэм в этот момент отлаживал машинный код, пошагово выполняя инструкцию за инструкцией. Он сказал: «Гляди, вот я собираюсь вызвать процедуру. Но как только я делаю шаг в машинный код – я оказываюсь не в этой процедуре, а на следующей строке после ее вызова». Я еле удержался от смеха. В архитектуре системы применялся «слот задержки», для того чтобы поддерживать заполненным конвейер инструкций, пока выполняется переход или вызов. То есть поведение, которое наблюдал Сэм, было нормальным явлением.

В другой раз Хелен пожаловалась мне, что компилятор как-то странно оптимизировал важную строку ее кода. Она пошагово выполняла машинный код, и в определенный момент ей нужно было выполнить вызов процедуры. Окно редактора кода отображало текущую позицию в соответствии с тем, какие машинные инструкции были выполнены. Она сказала: «Гляди, перед вызовом должно выполняться присваивание, а оно не выполняется!» А ведь наша оптимизация всего лишь передвинула это присваивание в слот задержки! Хелен удивилась бы еще больше, если бы сделала очередной шаг по коду, ведь ее процедура не начала бы при этом выполняться, зато произошло бы «потерявшееся» присваивание.

Эпилог

Разработка крупного комплекса программных инструментов, подобного HADS, может быть очень дорогим предприятием. Помимо приобре-

тения прав на передовую технологию компиляции, на проекте использовались внешние консультанты и инженеры из Honeywell. Хотя этих людей было немного, их работа стоила определенных денег. Некоторые руководители проекта очень противились любым расходам. На каком-то этапе реализации проекта появился компилятор Ада для процессора AMD 29050, разработанный сторонней компанией. Поскольку потенциальный рынок для такого компилятора и именно для этой целевой системы был очень невелик, весьма вероятно, что целевым заказчиком такого компилятора был проект 777. Возможно, кто-то в компании Honeywell поощрял поставщика компилятора к разработке этого комплекса средств, чтобы они могли заменить HADS или стать резервом в случае, если бы команда HADS потерпела фиаско. Однажды Брюс, один из наших менеджеров, подошел ко мне в коридоре и сказал: «Раз уже есть готовый компилятор, они просто прикроют проект HADS». Конечно, меня это очень тревожило, и даже не потому, что я боялся потерять работу, которой было много, но потому, что мы очень гордились качеством того, что сделали, и не хотели видеть, как результаты нашего труда пойдут на свалку. Но Брюс славился сплетнями и привычкой ставить других в неловкое положение, поэтому мы не стали к нему прислушиваться. Мы продолжали работать по плану. К тому времени, как вышел другой компилятор, команда HADS уже продемонстрировала значительные успехи. Общая оценка показала, что компилятор HADS однозначно лучше того компилятора.

Небольшая команда HADS добилась большого успеха, разработав крупную, качественную систему и ни разу не сорвав сроки. Конечно, перед командой проекта 777 у нас были некоторые преимущества. Например, есть стандарт, предписывающий, что именно должен представлять из себя компилятор Ада, то есть с самого первого дня проекта нам были известны базовые требования к продукту. Другим крупным преимуществом было то, что нам не нужно было создавать тома пользовательской документации, которые требовались в каскадном цикле проекта 777. Помимо документирования тех элементов среды выполнения, по которым требовалась сертификация DO-178B, мы создавали только внутреннюю документацию, которую считали необходимой для справочных целей, и только документацию для внешних пользователей. Это значительно облегчало жизнь команде HADS по сравнению с командой 777.

Но были и особенности команды, которые привели ее к успеху. Что способствовало превращению первоначально слабой команды в прекрасную, успешную команду?

Первым очевидным улучшением было то, что из проекта убрали некоторых людей, которые только создавали видимость работы. В результате получилась небольшая, но очень надежная группа. Очень неприятно работать с теми, на кого нельзя положиться. В большой команде можно долго не замечать, что кто-то ничего не делает. В маленькой группе всем очевидно, кто и чем занимается. Мы поняли, что гораздо лучше работать в небольшом коллективе надежных сотрудников. Кроме того, маленькая группа, выполнявшая львиную долю работы, была высокопрофессиональной. Многие из нас не работали прежде над компиляторами и сопутствующими инструментами, но все мы имели опыт работы в сложных проектах и приобрели очень хорошие навыки программирования в целом. Опыт в разработке именно компиляторов оказался не столь важен, важнее был опыт и навык применения нужных принципов проектирования ПО.

Команда отличалась не только надежностью и компетентностью. У нас очень развился коллективный подход. Мы все обладали определенными характеристиками, которые способствовали нашему успеху. И что самое важное, мы все знали, что каждый из нас очень хороший специалист. А поскольку мы это знали, нас не волновали личные похвалы или критика. Будучи уверены в своем профессионализме, мы не боялись просить других о помощи в особенно неприятных ситуациях. И именно поэтому нас не обижало, когда кто-то предлагал, возможно, лучшее решение проблемы. Зная, что мы профессионалы, мы готовы были тратить свое время на помощь другим. Зная это, мы признавали и профессионализм своих коллег. Помня о своем профессионализме, мы не считали необходимым настоять на своем во что бы то ни стало, но в решающие моменты могли быть очень настойчивыми. Некоторые говорили, что команда HADS была очень безличной, но я считаю, что в команде были очень яркие личности, однако им не нужно было постоянно подпитывать свое эго. Мы все были уверены в своих возможностях и в возможностях наших коллег.

Мы все очень упорно работали, каждый на пределе своих возможностей. Мы редко оставались сверхурочно, но каждый приходил вовремя и усердно трудился полный рабочий день. И так ежедневно. Поскольку мы всегда обращались за помощью к другим, когда у нас возникали проблемы, и видели, как упорно работают каждый день наши коллеги, мы всегда точно знали, что работа движется.

В ходе реализации проекта каждый разработчик в первую очередь отвечал за разработку в своей выделенной области. Например, у нас был менеджер по памяти, менеджер по исключениям, которые обеспечивали возможность компоновки в рамках требований AIMS по разделе-

нию приложений, назначению регистров и оптимизации. Поскольку мы знали, кто главный ответственный в определенной области, можно было пойти и обсудить проблему с нужным человеком.

Помимо компетенции и личного потенциала участников группы, мы применяли некоторые методы, которые привели нас к успеху.

Поскольку проект HADS разрабатывался самостоятельно внутри компании и специалисты проекта 777 хорошо знали разработчиков HADS, инженеры проекта легко могли обратиться к нам с просьбой о внедрении определенной функции или об оптимизации. Это могло привести к бесконечному «насаждению возможностей» или конфликту требований. Мы избежали этого, настояв на том, что все запросы должны проходить через Тома, лидера команды. Он принимал решение о том, над какими запросами стоило работать, и назначал приоритеты этим запросам. Он успешно оберегал всю команду от внешнего натиска. Уверен, что задача сохранения равновесия между защитой команды и внимательностью к требованиям проекта 777 была не из легких.

Очень полезной практикой было проведение коротких (получасовых) собраний по пятницам. На этих собраниях мы оценивали работу за прошедшую неделю и устанавливали общие приоритеты в работе на следующую неделю. Этот процесс был весьма демократичным, хотя последнее слово оставалось за Томом как за лидером проекта. На этих собраниях практически не решались общие проблемы, но нам давали высказать свои соображения по поводу возможного решения проблемы. Подобный подход сейчас применяется во многих Agile-командах, хотя в них приняты еще более короткие ежедневные собрания.

У нас были установлены этапы, очень жесткие по срокам и довольно жесткие по содержанию. На еженедельных собраниях определялось приоритетное направление работы на следующую неделю. Списки «топ 10» помогали нам сместить акцент на решение краткосрочных проблем, позволяющих достичь долгосрочной цели. Сочетание запланированных далеко вперед этапов, целей на следующую неделю и возможности вносить изменения ежедневно с помощью списков «топ 10», формировало гибкую среду разработки.

Разработка комплекса регрессионных тестов на основе ошибок, найденных либо самой командой, либо пользователями, обеспечивала постоянное подтверждение качества продукта. Также она облегчала процесс поиска регрессий, учитываемых в ходе разработки. Когда обнаруживалась регрессия, можно было оценить ее серьезность, чтобы определить ее приоритетность и разместить в чьем-нибудь списке «топ 10».

В проекте HADS было довольно приятно работать. Во многом это было связано с идеальностью самой команды и с успешностью проекта. Не-

просто понять, являются ли успех и те методы, что привели к успеху, причиной превращения команды в идеальную, или наоборот, идеальная команда способствовала успеху. Полагаю, это развивалось параллельно.

Каждый проект однажды завершается: либо провалом, либо выпуском ожидаемого продукта. Для команды HADS завершение проекта осуществлялось постепенно. Некоторых членов команды перекинули на проект 777 – работать в областях, где нужны были их экспертные знания Ада и близкое знакомство с комплексом программных средств. Постепенно ушли и консультанты, и в команде осталось два человека. На последнем этапе сокращений меня перевели в технологический центр Honeywell. Через несколько месяцев последний могиканин команды решил, что устал работать один, и покинул Honeywell. Поддержкой проекта занимались уже совсем другие люди. Ирония заключалась в том, что когда из команды HADS уже ушли основные разработчики, проект HADS был отмечен как лучший технический проект в Honeywell.

В этом рассказе изложена моя субъективная точка зрения. Не исключено, что другим участникам команды все представлялось в ином свете. Вероятно, хотя я в этом сомневаюсь, некоторые ключевые разработчики не считали команду идеальной. Но для меня работа в команде HADS стала одним из самых ярких периодов в карьере специалиста по программному обеспечению.

IV

Препятствия

При создании программ самые сложные проблемы создает то, чем невозможно управлять. Иногда проект проваливается из-за какой-нибудь случайности. Порой происходит что-то непредвиденное. А еще, к сожалению, иногда кто-то в команде нарочно старается провалить проект. Создание ПО – и без препятствий, встающих на пути команды, – дело непростое. Но мы работаем в реальном мире, и реальные проекты сталкиваются с проблемами, которые больше проблем только команды или программы.

Одно из самых главных препятствий – плохой руководитель.

Встречался ли вам руководитель, который отказывается утвердить правдоподобные оценочные данные? Или который не привык ценить ваши усилия («Команда говорит, что понадобится шесть недель? Я пообещаю уложиться в четыре, это им подпалит задницу!»), заставляя вас работать по выходным, чтобы успеть к нереальному дедлайну? Плохие руководители бывают самых разных сортов, и каждый из них представляет собой серьезное, иногда даже непреодолимое препятствие для команды. Порой кажется, что руководитель прямо-таки поклялся провалить ваш проект. Он руководит на микроуровне, постоянно меняет цели проекта, а когда дела пойдут плохо, остается неуязвимым, обвиняя в проблемах команду и никогда, даже ни на минуту не подвергая сомнению собственные методы руководства.

Работать с плохим руководителем бывает попросту невозможно. Но у команды, которой так не повезло, есть выбор. Иногда нужно просто понять, что руководитель работает против команды, и действовать в обход него. Иногда можно даже изменить плохого руководителя: если вы сможете завоевать его доверие, внушить ему уверенность команде, то все разрешится само собой. В любом случае, первый шаг состоит в том, чтобы осознать препятствие и понять, как с ним бороться.

Не всегда препятствие – это руководитель или другой человек. Иногда это просто какое-то техническое ограничение: необходимость работать с очень старым оборудованием, необходимость делать программы с обратной совместимостью; подчас мешают неожиданные, но серьезные проблемы с производительностью или непроходимый спагетти-код, который тяжело поддерживать.

Иногда проблемы – неизбежный результат того пространства, в котором работает команда. Например, может возникнуть проблема с помещением. Иногда команда вынуждена иметь дело с различными часовыми поясами – если она небольшими группами распределена по все-

му миру. Есть проблемы с погодой, с пробками, вообще с транспортом, с шумом... словом, всех проблем все равно не избежать.

И все эти проблемы могут потопить проект. Но если в вашей команде все друг другу доверяют и хорошо работают вместе, то вы сможете преодолеть даже самые безумные, неразрешимые на вид препятствия на своем пути. Более того, борьба с серьезными трудностями может сплотить команду. В конце концов, вы вместе. Если тяжелая проблема затрагивает каждого в команде, то, чтобы ее решить, все должны быть заодно. А когда цель достигнута, у всех появляется совместный опыт, который теснее связывает группу, и эти узы значительно крепче любых безоблачных мероприятий по тимбилдингу. Часто превратности судьбы только закаляют команду.

Говорить о препятствиях всегда тяжело. Преодолевать их болезненно, думать о них невесело. Более того, серьезные затруднения чаще всего действительно приводят к провалу проекта. Открыв любой учебник по программированию, вы найдете там разного рода статистику и исследования по поводу провала проектов. (Некоторые учебники приводят ужасающую цифру 80%!) Но многие ли из тех, с кем вы говорили, охотно поведали о своих неудачных проектах?

Никто не любит говорить о собственных ошибках. Большинство резюме подчеркивают успехи, часто стремясь найти достоинства и в неудачных проектах. Мало кто (если вообще хоть кто-нибудь) рассматривает собственную карьеру или трудовой опыт с точки зрения прогоревших проектов, и уж точно никто не считает ее серией неудач, одна за другой. Но если 80% (или даже вдвое меньше) проектов действительно завершаются провалом, то наши резюме и сами наши жизни должны пестреть ими.

Поговорив с действующими командами об их собственных проектах, мы именно это и обнаружили. Действительно, оказалось, что неудачных проектов очень много: проектов, которые существенно превысили смету или были сданы значительно позже срока; проектов, в результате которых появились неудобные программы, отвергнутые пользователями; проектов, в которых вообще ничего так и не было сделано. За эти годы мы не раз говорили о провале проектов, каждый раз спрашивая слушателей: «Есть здесь кто-нибудь, кто не участвовал хотя бы в одном провальном проекте?» Ни одна рука так и не поднялась.

Рассказы в этой части повествуют о командах, которые столкнулись с проблемами самого разного рода. Хорошие люди противостоят здесь злобным компаниям, ужасным руководителям, сталкиваются с серьез-

ными техническими проблемами и настоящим злом. Какие-то из этих проблем предотвратимы, но коварны и лишают веры в себя. От каких-то просто хватаешься за сердце. Какие-то неизбежны. И в каждом случае препятствие серьезно повлияло на команду – на их совместную работу и на то, как они это препятствие преодолели. Каждая история заключает в себе урок, который нужно усвоить, и рассказывает о людях, которые стали сильнее благодаря этому опыту, даже если команда и не устояла.

24

Плохой босс

Эндрю Стеллман

Дело было в середине 1990-х. В моде были фланелевые рубашки, мелодия из сериала «Друзья» прочно занимала верхние строчки хит-парадов, а я работал программистом в небольшой финансовой конторе, типичной для Уолл-стрит. Начав с младшего программиста, к тому времени я уже дорос до старшего разработчика. У меня была репутация того, кто может быстро выполнять крупные задания. Кроме того, у меня была репутация того, кто говорит, что думает, даже когда правильнее промолчать. Мой босс Питер однажды назвал меня «сорвавшейся с лафета пушкой, которую он научился разворачивать в нужном направлении». Меня это совершенно не задевало.

Интернет-бум еще не разразился в Нью-Йорке с полной силой, и компания, в которой я работал, была довольно консервативного и скучноватого толка. В таких компаниях слово «руководитель» многое значит. Только Начальнику с большой буквы Н полагается отдельный кабинет, и если кого-то назначают начальником, это большое событие. За каждое такое место просто дрались. В компании привыкли почитать боссов, а вокруг президента сложился целый культ личности. Рассказывали, что кого-то уволили на второй день работы только за то, что он назвал президента «стариком» в приватной беседе с коллегой, которая незамедлительно донесла об этом своему шефу.

Мне было наплевать на это. При всех этих обязательных воротничках, галстуках и общей консервативности я чувствовал, что занял отличную нишу. У меня была возможность работать с новейшими технологиями, и многое удавалось сделать. Питер доверял мне и не лез в мою работу, а со временем стал поручать все более важные и сложные задания. Я был уверен, что приношу компании большую пользу: разра-

ботанное мною программное обеспечение в значительной мере влияло на итоговый баланс. И у меня была отличная команда. Мы хорошо ладили, и я не припомню крупных ссор или скандалов между нами, по крайней мере до того момента.

Компания занималась подготовкой и поставкой инвестиционным банкам финансовых данных о добавленной стоимости. Большинство программ, которые я разрабатывал, были связаны с обработкой данных, получаемых нами от десятков поставщиков информации, составлением мудреных и таинственных финансовых расчетов, выверкой данных (обычно путем сравнения разных источников) и отправкой их нашим заказчикам, которые пользовались также специально разработанной программой, позволяющей работать с ними. Наша группа была лишь малой частью компании, большинство сотрудников которой были заняты тем, что вводили финансовые данные в разработанные нами приложения либо работали в отделе продаж, общаясь с клиентами.

Кроме меня в команде было пятеро. Три программиста: Пол, у которого была ученая степень по математике, он писал финансовые алгоритмы и низкоуровневый код; Дайана, очень талантливый разработчик графических интерфейсов и хороший программист, и Ленни, который работал в компании уже много лет, сначала в отделе ввода данных, создавая скрипты для других сотрудников отдела, а затем уже в нашей группе. Кроме того, было два тестировщика. Элен – великолепный специалист по техническому контролю: не единожды бывало, что, отдав ей «готовый» код, который считал безупречным, я получал его назад с огромным отчетом об ошибках для исправления. Младшего тестировщика Анну, самую юную в команде, Элен взяла под свое крылышко.

И все мы подчинялись Питеру. Он работал в компании с самого ее основания. Питер был мастер на все руки: протянул первую локальную сеть в компании, вел переговоры с поставщиками данных, занимался конфигурацией сетевых серверов, UNIX-машинами, брандмауэрами и маршрутизаторами и даже однажды провел полдня под дождем на крыше нашего офиса на углу Уолл-стрит и Бродвея, устанавливая микроволновую радиоантенну: одному из наших поставщиков данных она понадобилась, чтобы передавать нам биржевые сводки. Я очень уважал этого человека. Он работал больше, чем другие сотрудники, и никогда не боялся грязной работы. И, что еще лучше, он никогда не заставлял меня называть его «сэр».

Мы работали над крупным проектом уже три месяца. Это был важный для компании проект: программный комплекс должен был обеспечивать чтение портфелей акций наших заказчиков, подсчет прибыли по ним и сравнение с разными фондовыми индексами. Отдел продаж

очень ждал завершения нашей работы, поскольку ее результат должен был стать ключевой услугой, которую мы собирались предлагать нашим заказчикам. Команда обработки данных тоже очень ждала эту программу, поскольку она позволила бы автоматизировать целый ряд утомительных операций. Я руководил этим проектом и несколько недель общался с отделом продаж, группой ввода данных и старшим руководством, прежде чем мы смогли разработать рабочее задание, архитектуру и структуру базы данных. Дайана разработала пользовательский интерфейс, и вот уже я и Пол вовсю пишем код. Элен и Анна разрабатывали план испытаний и контрольные примеры, собирая довольно ценный набор тестовых баз данных, которыми мы все смогли бы пользоваться при написании и тестировании кода.

В целом проект двигался успешно, но не все было так просто. Мы разделили проект на три основных этапа, на которых должны были выпускать полнофункциональные компоненты окончательного продукта, и подходили к завершению первого этапа. (Позже я познакомлюсь с понятием «итерация». Но это была середина 1990-х, и лучшее, что у нас было на тот момент, – это поэтапный выпуск продукта.) Итак, мы завершали первый этап. Это была часть программы, с которой сотрудники отдела обработки данных могли уже работать, пока мы проходим остальные два этапа. А ее небольшой фрагмент наши специалисты по продажам предполагали демонстрировать клиентам, чтобы подогреть их интерес к будущей покупке.

Пришло время продемонстрировать результат нашей работы, и я собрал пятерых своих коллег в конференц-зале, попросив Питера пригласить Аньес, вице-президента группы ввода данных, главу отдела продаж (который привел двух лучших своих специалистов просто потому, что они в тот день оказались в офисе) и президента. Я с гордостью продемонстрировал разработанное нами ПО, указывая им на его достоинства и недостатки. Мы все с большим волнением ожидали их реакции на результат нашего труда. Пока не *дождались*.

После презентации воцарилась тишина. В конце концов, Аньес сказала «Пожалуй, выглядит неплохо. Уверена, что мы сможем ее применить». По голосу было понятно, что мы ее не убедили.

«Но?...» – спросил я.

«Но разве не предполагалось, что она будет получать данные клиентских портфелей по сети?» – последовал вопрос.

Так. Я понятия не имел о том, что предполагалось получать данные клиентских портфелей по сети. Очевидно, в своих обсуждениях с ними я выпустил из виду этот аспект. Это была катастрофа, и исключительно по моей вине. Вся моя команда старательно отводила глаза.

Совещание закончилось в три сорок пять. Я увел команду пить кофе за мой счет до конца рабочего дня. Нам нужен был план. Мы спорили. Мы последовательно прошли через отрицание, гнев, торг и приятие. Мы разработали план: нужно собрать всех заинтересованных лиц, чтобы они точно определились с тем, что необходимо разработать. Мы будем работать с ними вместе и по отдельности. Мы запишем все, что нам скажут, и попросим проверить эти записи всех участников. Мы исправим недостатки в программе. И оставшиеся элементы мы будем разрабатывать уже без ошибок.

Так все и пошло. Нам повезло – и люди из отдела данных, и специалисты отдела продаж хорошо знали, что им нужно. Им действительно нужна была хорошая программа, и они готовы были целый день обсуждать со мной свои задачи и проблемы. Начав все заново, я постарался понять, что им на самом деле нужно. Я задавал огромное количество вопросов.

И я научился на этом горьком опыте. Теперь я уже не занимался этим в одиночку. Нет, теперь вся команда участвовала в процессе – как только возникал вопрос о том, как пользователи будут работать с этим ПО, Дайана спешила разъяснить, как будет устроен интерфейс. Ленни хорошо знал поставщиков данных и применяемые ими форматы; кроме того, он иногда выступал в качестве переводчика (поскольку я говорил на языке программистов, а пользователи – на языке отдела данных). Пол замечательным образом разобрался в расчетах, которые мы должны были обеспечить. Элен и Анна настаивали на том, чтобы присутствовать на большинстве наших совещаний. Я не очень понимал, зачем им это, но они очень много всего записывали и брали образцы всех файлов данных, с которыми нам предстояло работать. (В итоге они выдали нам полную тестовую базу данных, что сильно помогло разработчикам; кроме того, они сделали очень подробный набор контрольных примеров.)

Две недели спустя казалось, что все наихудшее уже позади. Специалисты отделов продаж и обработки данных были замечательными людьми – они как минимум трижды прошли с нами перепроверку всех требований, что, должно быть, далось им нелегко. Но у меня сложилось четкое ощущение, что мы на верном пути.

И, что еще важнее, мы снова двигали проект вперед. И мы вшестером снова были в прекрасных отношениях. На самом деле, я по очереди спросил у каждого члена команды, не злится ли он на меня. «Вовсе нет», – сказала мне Анна. И в самом деле, после того как я поговорил с каждым, у меня возникло чувство, что мы действительно очень тесно связаны друг с другом. Мы прошли испытание огнем и вышли из него с честью.

Единственная проблема была в том, что Питер все больше становился помехой для нас, и он сам это понимал. Нам приходилось звать его почти на все собрания. Без него нельзя было принять многие решения, особенно когда речь шла о том, чтобы привлечь людей к работе на проекте, отвлекая их от другой работы. В какой-то момент он отвел меня в сторонку.

«Слушай, – сказал он. – Я только что принял четырех новых ребят в группу технической поддержки и должен уделять им много времени».

Мы обсудили, что еще требуется нашей команде. Он понимал, что большей частью его работа сводится к «посредничеству» между нами и другими руководителями. Выяснилось, что у него уходит много времени на это «решение чужих проблем», на защиту нас от постоянной канонады запросов на внесение изменений, поправок и реализацию новых проектов, которые, возможно, были и не нужны. Но в то время все большего напряжения от него требовали как задачи программирования, так и общие задачи информационных технологий. Компания росла, и ему поручили разработать новый центр сбора и обработки данных, чтобы разместить нашу все возрастающую сетевую инфраструктуру.

Настал момент, когда Питер сказал, что нам нужен менеджер на полную ставку. Нужно разделить ИТ-поддержку и разработку. Он бы напрямую работал с отделом ИТ, а команду разработки возглавит кто-то другой.

Он спросил меня, не хочу ли я занять эту должность.

Я никогда всерьез не задумывался о руководящей работе и знал о ней только то, что она меня не очень интересует. «Знаешь, очень приятно, что ты предложил мне это. Но, по-моему, у меня просто нет необходимого опыта».

Он огорчился, но не стал на меня давить.

«Тогда я начну подыскивать вам нового руководителя», – сказал Питер.

В течение следующих полутора месяцев Питер провел не меньше десяти собеседований. И почти всем кандидатам отказал. Видимо, найти хорошего руководителя непросто. Однажды он пригласил меня к себе в кабинет.

И сказал: «Знакомься, это Эрик».

Передо мной сидел очень большой дядя, под два метра ростом, почти полностью лысый, в костюме-тройке, и поглядывал на свои супердорогие часы. Булавка в галстуке и запонки идеально сочетались. Судя по его виду, он определенно вписывался в корпоративную культуру нашей компании, даже лучше, чем я. Он переходил к нам из большого инве-

стиционного банка – как выяснилось, он был вице-президентом нашего крупнейшего клиента. У него есть опыт работы с большими командами разработчиков, заявил он, и с нашей маленькой командой он точно справится. Похоже, Пол нашел подходящего руководителя!

Все мы очень волновались. Сразу после знакомства с Эриком я встретил Дайану, и она очень обрадовалась его приходу. Оказалось, что Питер уже познакомил их, и Эрик сказал ей, что ценит ее работу и думает о том, чтобы повысить ее и дать ей большие полномочия в команде (чего ей очень хотелось). Потом я пообщался с Полом, который сказал мне, что Эрик слегка намекнул ему, что при следующем пересмотре Пола ждет большое повышение. В целом, все мы были рады, что кто-то придет и избавит нас от повторения проблем, решение которых отнимало целые недели у нашего большого проекта.

Спустя две недели Эрик вышел на работу. Ему отдали кабинет Питера, а сам Питер переехал в кабинет побольше и поближе к своим ИТ-специалистам. Эрик привез с собой кубки гольф-клуба и несколько грамот с предыдущей работы в инвестиционном банке. Установил на своем столе фотографию большой яхты – по всей видимости, его собственной – в массивной деревянной рамке со встроенными изящными морскими часами. Теперь этот кабинет стал кабинетом Начальника с большой буквы Н.

Эрик устроил в конференц-зале общее собрание нашей команды.

«Я очень уважаю хороших разработчиков, – сказал Эрик. – А вы очень хорошая команда разработчиков. Я признаю и уважаю это».

Пока все шло неплохо. Он сообщил, что собирается «не стоять у нас на пути, дав нам делать свое дело». Что же, хорошо. Потом он немного поговорил о кораблях и о том, как хороший капитан держит штурвал, а команда выполняет свою работу. Он говорил о навыках высокоэффективных людей. О доверии и о том, что хочет завоевать наше доверие.

Затем он рассказывал о тренинге для руководителей в его банке, когда вице-президенты бегали по лесам под руководством бывшего морпеха. Ладно. Я не вполне понимал, к чему он ведет. Я никогда не был поклонником обязательных корпоративных тренингов, вроде «доверяющего падения» и «давайте обнимемся», а Эрику они, очевидно, нравились. Но я все же оставлял ему презумпцию невиновности. Я посмотрел на коллег, чтобы понять их реакцию. Дайана и Пол, похоже, уже купились на все это. Элен и Анна выглядели менее воодушевленными.

В завершение встречи Эрик сказал, что нас ждут перемены. «Я не такой, как Питер, – сказал он. – Я докажу, что уважаю вас и готов за вас драться». Это звучало несколько странно. У меня никогда не возникало чувства, что Питер относится к нам иначе, чем с уважением, и я поня-

тия не имел о том, что Питер в свое время не отстаивал наши интересы (или что ему нужно было это делать). Ленни выглядел несколько отрешенным. Пол, обычно умеющий хранить хорошую мину при плохой игре, хмурился.

После встречи я решил поговорить с Элен. У нас с ней были очень хорошие рабочие отношения; последние два года мы нередко подолгу обсуждали возможности улучшить нашу работу. Многие собрания, проведенные мной в рамках исправления ошибки в проекте, на которые я приглашал специалистов по вводу данных и по продажам, были организованы по ее подсказке и сыграли очень важную роль. Кроме того, Элен замечательно разбиралась в людях, поэтому мне было интересно, что она скажет.

«Если честно, пока я не очень рада, – сказала Элен. – Ты заметил, что он ни разу не обратился ко мне или к Анне? Он говорил только о программировании. По-моему, у нас с ним будут проблемы».

Я совершенно не придал значения этим словам. (Теперь я знаю: нельзя пропускать мимо ушей то, что говорит Элен.) И сказал: «А я думаю, что он хороший руководитель. Надо дать ему шанс». По какой-то причине я порой излишне оптимистично отношусь к человеку, особенно если этот человек – мой руководитель. И это не последний случай того, как этот оптимизм мне аукнулся.

Позже на той же неделе Эрик пригласил меня в свой кабинет. Он усадил меня и ухмыльнулся. У него была обезоруживающая ухмылка.

«Я разговаривал с высшим руководством компании. Им нужны две вещи, обе первостепенной важности. Первая – это интранет».

«Мм, Эрик, я не совсем понял, о чем ты». У нас уже была внутрикорпоративная сеть, отделенная от Интернета брандмауэром. (Напоминаю, что была только середина 1990-х, поэтому у нас был канал только на 56 К.) У нас было несколько веб-серверов, на которых были запущены некоторые внутренние приложения на ранней версии IIS; кроме того, у нас было несколько файл-серверов коллективного доступа. Что еще могло нам понадобиться?

«Я говорил с топ-менеджерами – они хотят большего. Им нужно полномасштабное интранет-приложение, единая исходная точка для всех наших внутренних приложений. У него должна быть обширная система меню. Оно должно быть полностью настраиваемым. Оно должно давать доступ ко всем нашим внутренним ресурсам: каждой программе, каждому файлу, каждому элементу информации. Плюс броузер баз данных, подсистема просмотра файлов и форум. И они ждут, что ты это разработаешь».

Ладно.

«А второе что?» – спросил я.

«Второе – необходимо расширить команду. Тут мне пришлось побороться, но я убедил их добавить должность еще одного разработчика».

Я не очень понимал, зачем ему нужен дополнительный разработчик. У нас был не такой уж большой завал работы. У каждого было достаточно работы, но никто и не требовал от нас, чтобы мы разработали ПО дополнительно. На самом деле, мне казалось, что у нас прямо противоположная проблема: нам было очень сложно привлекать к работе специалистов по продажам и по данным, отрывая их от работы. Мы уже кучу времени потратили в ожидании, пока они освободятся. Мне казалось, именно поэтому Питер и хотел найти для команды нового руководителя.

Однако Эрик считал иначе: «Я сделал ставку на вас, ребята, на команду. Я поднял большой шум наверху и после борьбы таки добился, чтобы они добавили еще одну позицию программиста. Вот как я забочусь о своей команде».

За следующие несколько дней я часов пять обсуждал с Эриком вопрос интранета. Поначалу казалось, что он довольно хорошо представляет, чего хочет руководство. Около часа он говорил о «великом плане». Я не все понимал, но многое записывал. Он говорил о меню, об опциях, базах данных, броузере данных и еще целой куче вещей, для воплощения которых мне потребовалось бы не меньше трех месяцев.

Я занялся этим проектом. Но вернувшись к тем записям, понял, что «великий план», возможно, и был хорош, но конкретный план не продуман. И я снова пошел к Эрику выяснять детали. Например, меню – что значит «меню с перестраиваемой конфигурацией»? Хотел ли он, чтобы можно было конфигурировать меню при помощи файлов или баз данных? Нужен ли был для этого пользовательский интерфейс или достаточно было напрямую редактировать базу данных и файлы?

Ответ Эрика меня совсем не устроил: «Ты уполномочен принимать такие решения самостоятельно». А потом у него был срочный телефонный разговор. Я не вполне понимал, что значит «уполномочен». Мне не нужна была возможность решать, нужны были ответы. У Эрика их не было. Следующие два дня он не отвечал на мои электронные сообщения. И я начал разрабатывать интранет на основе своих записей и соображений. Так я работал несколько недель. К счастью, я очень изобретателен и сам придумал, как может выглядеть внутрикорпоративная сеть. И все это время я не получал практически никаких указаний от Эрика – каждый раз, когда я разговаривал с ним, он, похоже, даже не понимал толком, над чем я работаю. Но когда я показал ему результаты моего труда, он сказал, что очень рад. Я многое сделал и разрабатывал именно то, о чем он меня просил.

Что же, он был боссом. И он был мной доволен. А я делал свою работу... так?

Прошло около трех недель с тех пор, как Эрик стал нашим начальником. Я работал над созданием какого-то элемента внутренней сети, когда ко мне зашла Дайана.

«Ты уже разговаривал с Эриком?» – спросила она.

Выяснилось, что я хоть как-то взаимодействовал с Эриком, а с ней он лично ни разу не поговорил. Завершив разработку пользовательского интерфейса для нашего большого проекта (еще задолго до прихода Эрика), она начала работать над проектом для Аньес, вице-президента группы ввода данных. Так они и работали с Анной над тем проектом, а спустя две недели после прихода Эрика завершили работу над ним.

За последнюю неделю команда Аньес обнаружила пару крупных ошибок, и Дайана их исправила. Но кроме этого она ничем особым не занималась. Она отправила Эрику пару сообщений о том, что готова приступить к новому проекту, но он не ответил ни на одно. Она оставила ему голосовое сообщение, которое он также проигнорировал.

Я подумал, что это очень странно. Несомненно, работа для нее была. Я работал над интранетом, поэтому не мог заниматься тем крупным проектом. На нем остались Пол и Ленни, и они, уж конечно, нашли бы, к чему ее приспособить. Что было бы очень хорошо, поскольку она и сама хотела работать там. Проект был очень серьезным, и она уже вложила немало сил в разработку графического интерфейса. Но когда она в последний раз разговаривала с Эриком, случайно столкнувшись с ним в коридоре пару недель назад, он сказал ей «ни в коем случае не вмешиваться в проект». С тех пор она не получала от него никаких инструкций.

Я сказал ей: «Не волнуйся. Думаю, тут какая-то неразбериха. Он, конечно, включится в работу, когда мы разъясним ему ситуацию».

Я был очень наивен. Отправил Эрику большое письмо с объяснениями. Рассказал ему обо всей работе, которую уже выполнила Дайана, о ее незаменимом вкладе в большой проект, о том, что она готова приступить хоть сейчас. Я ясно давал понять, что питаю к ней большое уважение и что имеет смысл немедленно пустить ее в дело.

Никакого ответа. Тишина.

Я пришел к его кабинету и постучался. Его не было на месте. Очевидно, он неспешно обедал где-то со своими старыми приятелями из инвестбанка. Я поспрашивал позже и выяснил, что он сказал одному из специалистов по сбыту, что у него, возможно, получится вывести нашу компанию на выгодное положение в отношениях с банком. Это не было

его обязанностью, но я подумал, что раз это хорошо для компании, то пусть занимается... так?

Он вернулся без пятнадцати четыре. Я начал рассказывать ему про Дайану.

«А, да, я видел сообщение. Не волнуйся, я займусь этим».

Отлично! Но что именно он собирался делать?

«Я же сказал: не волнуйся об этом», – он выглядел взбудораженным и хотел сменить тему. Потом он передал мне толстую пачку бумаг. Там было около 50 резюме. «Нужно найти человека на то место, – сказал Эрик. – Вот пачка кандидатур. Просмотри, обзвони всех и выбери лучших». Прежде мне не приходилось изучать резюме в таком масштабе. И мне казалось, что это *его* работа. Я же не просто так отказался от должности руководителя, когда Питер предлагал ее мне.

«А как же проект интранета?», – спросил я.

«Это сейчас важнее», – ответил он.

Я решил не поднимать вопрос Дайаны. Мне стало неловко из-за этого, но я понимал, что тут ничего поделать не могу. Я вернулся к ней и рассказал о том, что произошло. Она отреагировала нормально, но попрежнему не знала, что ей делать. Был один элемент, за который пока не брались, и я знал, что с этой работой она справится. Она уже разработала графический интерфейс для него и хорошо представляла себе, как должны работать компоненты. Да, Эрик сказал, что позаботится об этом. Но он не сказал, что именно собирается делать, и у меня возникло чувство, что он хочет бросить Дайану на произвол судьбы. Я уже начинал подозревать, что Эрику, возможно, на самом деле наплевать на нас. Поэтому я сказал Дайане приступать к работе над большим проектом.

«А что насчет Эрика?», – спросила Дайана.

Я сказал, что возьму это на себя. «Если он поднимет шум, я возьму вину на себя. Скажу, что неправильно его понял», – применил я одно из своих любимых правил (которое, между прочим, перенял у Питера): иногда лучше извиниться завтра, чем просить разрешения сегодня. Я поговорил с Полом и Ленни, и вместе мы определили для Дайаны участок работы в рамках большого проекта. Они были очень рады такой помощи. Может, Эрику и было наплевать на проект, но команде – нет. Они были счастливы видеть ее в своих рядах.

После той встречи Ленни отвел меня в сторонку: «Слушай, нужно, чтобы ты поговорил с Аньес».

Аньес подходила к Ленни как минимум раз в день с просьбами помочь в поддержке старых скриптов, над которыми он когда-то работал. И это было проблемой. Когда два года назад Ленни официально перешел из

ее отдела в команду Питера, Аньес пообещала, что ему не придется заниматься теми старыми программами. У них был человек, который мог отлично справляться с этой работой, а Ленни нужно было работать над нашим большим проектом. Но Аньес хотела, чтобы тот человек занимался «более важными» делами, поэтому она продолжала обращаться к Ленни по поводу тех скриптов.

Это была одна из задач «разруливания», для решения которых как раз и взяли Эрика. Но когда Ленни сказал ему об этом за пару дней до нашего разговора, Эрик заявил, что он уверен, что Аньес перестанет это делать. Он пообещал Ленни, что «задаст ей жару», и больше она так делать не будет. Однако вчера Аньес снова пришла к Ленни с просьбой об очередной модификации скрипта.

Что-то определенно шло не так. За несколько часов до этого я встречался с Аньес, и она выглядела вполне как обычно – самодовольной, чересчур самоуверенной и в целом довольно невыносимой – той самой старой Аньес, какой мы ее знали и любили уже много лет. Если бы Эрик «задал ей», как обещал, она была бы раздраженной и злой. (Насколько я знаю, у Аньес было только два этих состояния, заметных издалека.) Вместо ворчания она, наоборот, принялась расхваливать успехи Эрика. Что-то здесь не сходилось.

Я пообещал Ленни, что поговорю с Аньес и попробую ее остановить. Это будет не так сложно сделать: ей нужно было завершение большого проекта, и, как я полагал, она просто не понимает, что Ленни во многом определяет успешность этого проекта. Я думал, что если подам это в таком свете, она согласится отступить. И когда я пошел к ней, именно в таком русле и произошел наш разговор. Ленни был однозначно счастлив, что с него сняли бремя поддержки старых скриптов, а мы вернули себе программиста.

К тому времени я проводил уже больше половины своего времени, занимаясь наймом программиста на должность, которую выбил для нас Эрик. Моя работа над интранетом значительно замедлилась, поскольку я много времени тратил на звонки в кадровые агентства и на телефонные беседы с кандидатами для Эрика. Помимо этого, я выдавал задания команде, решал вопросы с другими руководителями и в целом выполнял работу Эрика. Оставалось только удивляться, чем же занимается сам Эрик.

Прочтя десятки резюме и поговорив более чем с десятью кандидатами, я наконец нашел программистку, идеально подходившую для этой работы. И принес ее резюме Эрику. Она действительно была идеальным кандидатом: 15 лет работы программистом в консалтинговой компании, а до этого еще 10 лет аналитиком в инвестиционном банке. Она

знала язык наших программ и несколько лет разрабатывала финансовые программы.

Он посмотрел на резюме. «*Она* закончила университет в 1961 году. Значит, ей сейчас уже за 50?» – он так надавил на слово *она*, что мне стало неуютно.

«Думаю, что да, – ответил я. – Я не спрашивал, сколько ей лет».

«Хм, – сказал Эрик. Он еще немного пробежался по резюме. Я видел, как минутная стрелка его морских часов в фоторамке перепрыгнула на следующее деление. – Я позвоню ей и сообщу тебе, как все прошло».

Я вернулся на рабочее место. Через пятнадцать минут Эрик снова вызвал меня.

«Она нам не подходит».

Я попросил его обосновать это. Он отказался. «Это мое решение. Я не вполне уверен в некоторых ее, так сказать, *характеристиках*». Я попросил уточнить. Он ответил, что в общем ему тут больше нечего сказать. «Понимаешь, все дело в ее возрасте и в том, что она женщина. Это не очень нам подходит».

Я был потрясен. Знаю, нужно было сказать ему, что совершенно незаконно, кроме того что очень оскорбительно лично для меня, отвергать лучшего кандидата только на основании того, что это женщина и ей за пятьдесят. Но я совершенно растерялся. Я вышел из его кабинета, позвонил в кадровое агентство, сказав, что мы вынуждены отказаться от этого кандидата, и уехал домой, не дожидаясь конца рабочего дня.

На следующий день Эрик попросил меня выделить время после обеда. Он провел собеседование по телефону с одним из следующих отобранных мною кандидатов, и теперь тот должен был прийти на личное собеседование. Я подумал, что это довольно странно, потому что я не говорил Эрику ничего о следующих кандидатах. Он передал мне резюме. Не помню, чтобы я видел его прежде. У этого парня было гораздо меньше опыта, но в целом он вполне мог подойти.

Собеседование назначили на час дня. В пятнадцать минут второго мне позвонила секретарь из приемной и легкомысленным тоном сообщила что-то вроде «тут к вам Элвис пожаловал». Она всегда подшучивала и хихикала. Я прошел к выходу, и там, действительно, стоял Элвис. Не то, чтобы тот самый молодой Элвис – этот был весьма упитанным,– но и не сказать, чтобы толстый старый Элвис. Волосы, уложенные в небольшой кок, длинные баки (если помните, была середина 1990-х, и благодаря Люку Пери в сериале «Беверли Хиллз 90210» бакенбарды снова вошли в моду) и темные очки.

Я отвел Элвиса в переговорную и побеседовал с ним. В резюме он несколько преувеличил свои умения, но в целом произвел впечатление человека, который хорошо понимал, что делает. Он уже занимался прежде программированием, большей частью в области баз данных. Пожалуй, он сможет поддерживать сценарии, написанные когда-то Ленни, то есть работа для него найдется. Закончив собеседование, я написал Эрику, что рекомендую этого человека на должность помощника программиста.

На следующий день все члены команды получили от Эрика сообщение о том, что он только что принял на работу старшего программиста. Он был рад уведомить нас, что этот старший программист возьмет на себя большую часть нагрузки по большому проекту, над которым постоянно работали Пол, Ленни, Дайана, Элен и Анна. И этот новый старший программист... неужели, Элвис?!

Нет, не может быть, здесь какая-то ошибка. Он не мог быть старшим по определению. И абсолютно точно он не смог бы справиться с такой задачей. Он мог бы заниматься поддержкой простых скриптов, возможно, делать что-то по базам данных. Но у него не было ни навыков, ни опыта, чтобы взять на себя любой из элементов большого проекта. Я пытался разыскать Эрика, но он снова был на затянувшемся ланче. К тому времени, как он вернулся в офис, рабочий день почти закончился. Я сказал ему, что Элвис точно не справится. Он ответил, что нужно верить в людей. Моя вера начинала однозначно трещать по всем швам.

Две недели спустя Элвис появился в отсеке напротив моего. Он пришел поздно, последним из команды. Он так и не снял очки. Он поболтал с секретарем в приемной. (Позже она рассказала мне, как он пытался с ней заигрывать, придя на собеседование.) Мы собрали совещание команды проекта после обеда. Пол и Ленни разработали план ввода Элвиса в курс дела. Но Элвис так и не появился.

На следующий день он сказал Полу, что у него было срочное личное дело и ему пришлось убежать пораньше. Пол очень разозлился – он все утро готовился к этому собранию, вместо того чтобы писать код. Они начинали отставать от графика по большому проекту и нужно использовать время по максимуму.

Спустя два дня Элвис вовсе не появился на работе. Не пришел он и на второй день. Эрик молчал. Он вел себя так, будто ничего не происходит. Позже в тот день я встретил в лифте секретаря, и она рассказала мне, что случилось.

«Элвис не прошел тест на наркотики, и нам пришлось с ним расстаться», – сказала она.

В компании была политика проведения таких тестов в отношении всех новых сотрудников. Каждому кандидату сообщали об этом на первом телефонном собеседовании. Элвис знал о тесте, и у него было две недели на то, чтобы не принимать наркотиков и пройти его чистым.

«Погоди минутку, – сказал я. – Помнится, у одного из наших айтишников нашли следы травы и спустили на тормозах».

«Да, но у Элвиса нашли кокаин. Тест показал, что он принимал его в течение предыдущих двух дней. Он здесь больше не работает».

Не скажу, чтобы его уход меня сильно огорчил. Я никому не рассказал о том, что случилось, но позже в тот же день услышал эту историю как минимум от трех участников нашей команды. Похоже, все знали про Элвиса и его проблему с наркотиками.

Эрик воспринял все это очень спокойно. Он передал мне еще пачку резюме и попросил позвонить как минимум десяти из этих кандидатов. Ситуация становилась смехотворной. Я не работал над проектом интранета уже неделю, а Эрика это совершенно не волновало. И я решил поговорить с Питером. Обычно я так не поступаю, не обращаюсь к высшему руководству через голову своего начальника, но я очень долго работал с Питером. Я чувствовал уверенность в необходимости этого разговора и хотел получить прямой ответ.

«Какой еще проект интранета?» – спросил он.

Тут и выяснилось, что никто в компании понятия не имеет о том, что я работаю над этим проектом. Я не поверил своим ушам: Эрик сказал, что это жизненно важный проект, особый запрос высшего руководства. Питер присутствовал на всех стратегических совещаниях, где высшее руководство обсуждало новые проекты, и этот вопрос никогда не ставился. На самом деле, Эрик ни разу *не был* на таком совещании, поэтому не мог знать, чего хочет высшее руководство. Очевидно, Эрик самостоятельно установил этот приоритет, а я провел месяцы, работая над проектом, который больше никому в компании не был нужен.

Я вернулся к Эрику, который был искренне раздражен тем, что я не обзваниваю кандидатов на замену Элвису. Он и слышать не хотел о проекте интранета. Его не интересовал ход работы, если честно, похоже, он начисто забыл о его существовании. Я сказал ему, что мог бы быть очень полезным Ленни, Дайане и Полу на большом проекте.

«Слушай, что ты беспокоишься. Ты делаешь большую работу, – сказал он мне. – Оставь большой проект остальной команде. У тебя есть дела поважнее».

Я хотел спросить, что он имел в виду под этими важными делами – проект интранета или активную дискриминацию кандидатов по полу и возрасту. До сих пор жалею, что у меня тогда не хватило на это *пороха*.

Я вернулся за свой стол, запоздало сожалея об упущенной возможности наконец-то высказаться. И тут я услышал громкий крик Эрика. Я заглянул в кабинет – дверь была открыта – и был просто шокирован, увидев, что кричит он на Элен и Анну. Из того, что я услышал, было понятно, что речь идет о какой-то критической ситуации с данными клиентов.

По завершении экзекуции Элен и Анна вышли. Анна тихо всхлипывала у себя за столом. Элен выглядела так, будто готова выбить кому-то все зубы. Выяснилось, что проблема была с некоторыми данными, которые использовал крупный клиент. Этот клиент пригрозил закрыть счет. При этом и Элен, и Анна точно знали, что они не имеют отношения к этой проблеме. И не потому, что они тестировщики, дело было даже не в том, что они упустили какую-то критическую ошибку. Проблема возникла у одного из поставщиков данных. Мы абсолютно ничего не могли сделать по этому поводу. Если уж кого и можно было обвинить, так это Аньес и ее группу ввода данных, которые упустили это проблему из виду. Но Аньес очень хорошо умела избегать любых неприятностей, поэтому она свалила вину на Эрика. А Эрик выбрал Элен и Анну в качестве козлов отпущения.

Мы с Элен позвали Анну выпить кофе в соседнее кафе. Она была сама не своя. Весь офис слышал, как Эрик орал на нее, ведь он оставил дверь открытой (видимо, специально, чтобы еще больше смутить их). Мы успокоили ее, но больше она в тот день работать не могла. В конце концов, она взяла отгулы на следующую пятницу и понедельник и поехала в маме. По возвращении во вторник она все еще выглядела расстроенной, но по крайней мере могла работать. Я спросил, в порядке ли она.

«Да, Эндрю, – сказала она. – В общем, в порядке. Я хочу завершить большой проект. Я уже очень много в него вложила. Но когда мы закончим, дадите мне рекомендацию, чтобы я могла найти новую работу?»

Я сказал, что обязательно дам. Я был рад, что она хотя бы останется с нами до конца проекта.

А у нее были хорошие результаты. Вся команда очень быстро двигалась вперед. Как ни удивительно, вся команда искренне болела за дело завершения большого проекта. Мне очень хотелось быть с ними, а не делать за Эрика его работу. К настоящему времени я один не был активно задействован в том проекте. Вся остальная команда выполнила уже два этапа из трех, оба отдела-заказчика нашего проекта одобрили то, что им показали. Аньес даже прислала им сообщение с поздравлениями, что было для нее нехарактерно. Они были очень счастливы – так счастливы, что им почти не испортил настроение высокомерный ответ Эрика (с копией всем сотрудникам компании), где он приписывал этот успех себе.

Компания расширялась. Когда я только пришел работать, мы занимали в здании примерно половину седьмого этажа. Теперь мы уже заняли весь седьмой этаж и подписали договор на аренду половины восьмого. Для нашей команды это была удача. Они начали собираться в переговорной на восьмом, в результате чего Эрик, которого не устраивало, что приходится тащиться по лестнице, перестал заявляться на их собрания без предупреждения и с длинными рассказами о своей бытности вице-президентом в инвестиционном банке.

Позже от Пола я узнал, что его двоюродный брат работал в том же инвестиционном банке, где и Эрик. Он сказал, что слышал от брата, который тоже был вице-президентом, что они раздавали эту должность направо и налево. Идея, видимо, была в том, что когда звонил мелкий клиент откуда-то со Среднего Запада и просил соединить его с кем-то, всегда можно было соединить его с «вице-президентом таким-то», который будет лично вести его счет. Это очень эффективный инструмент продаж, поскольку мелкий клиент любит, чтобы с ним обращались, как с крупным воротилой.

Это многое объясняло. Эрик уже не казался столь внушительным. Если верить брату Пола, тот факт, что он был вице-президентом, означал лишь, что он не был уборщиком.

(Если честно, я стал подозревать, что уборщик нашего офиса работал лучше Эрика. Я несколько раз беседовал с ним, когда задерживался, работая над проектом. Несмотря на тюремные наколки, это был приятный парень. Его звали Майк. К сожалению, здание потом перешло к компании по уборке, не входившей в профсоюз, и он ушел. Профсоюз работников сервиса устроил забастовку возде нашего здания, мне было как-то неудобно проходить сквозь их пикет перед входом, и я начал пользоваться запасным.)

Примерно в то же время Элен должна была пройти ежегодную аттестацию. Она не знала точно, чего ожидать, поскольку единственный раз, когда она пересекалась с Эриком лично, был тот, когда он на нее кричал. Однако она хороший профессионал, а на последних встречах с Питером ей обещали повышение. К сожалению, придя на аттестацию, она увидела то, что не предназначалось для ее глаз. Эрик уже написал отчет по моей эффективности и оставил его неперевернутым на столе. Внизу крупным шрифтом была указана моя зарплата.

Она была примерно вдвое выше, чем у нее.

Элен проработала в компании примерно столько же, сколько и я. У нее было больше опыта, чем у меня, поскольку она проработала в разработке программного обеспечения больше лет. Да, известно, что тестировщики получают немного меньше программистов. Но она действительно

была ценным работником для компании, и для нее было унизительно получать вдвое меньше, чем я. Это было просто неприемлемо. Элен не знала, как поступить, поэтому взяла тот отчет, перевернула его и отдала Эрику.

Позже она поговорила с Эриком.

«Послушайте, – сказала она. – Я не собиралась выяснять, сколько зарабатывает Эндрю. Но теперь я это знаю. Мы оба понимаем, что несправедливо, если он будет получать вдвое больше, чем я, и я надеюсь, вы примете правильное решение». Она дала ему понять, что хочет зарплату не как у меня, но хотя бы того же порядка.

Позже в тот же день Эрик сообщил ей плохую новость. Он не собирался поднимать этот вопрос перед высшим руководством. И Эрик напомнил, что все мы видели статистику, по которой женщины получают меньше, чем мужчины, то есть такая картина не должна ее удивлять.

Через неделю Элен написала заявление и ушла из компании.

Эрик был просто счастлив. Теперь в его группе было две вакансии: одну он якобы выбил для нас, а теперь появилась еще одна – на замену Элен. Он выдал мне следующую пачку резюме.

Я пролистал их, заметив, что среди них нет ни одного тестировщика или инженера по качеству, только программисты. И спросил об этом Эрика.

«Разве мы не собираемся искать замену Элен? Иначе Анне придется взвалить на себя всю нагрузку по большому проекту».

Эрик и слышать об этом не хотел. Он заявил, что не нуждается в тестировщиках и что один программист стоит пятерых таких специалистов. Сказал, что, на его взгляд, в нашем деле вообще не требуются тестировщики, что с этой работой справится стажер или даже смышленый подросток.

Я ответил, что, пожалуй, несколько опрометчиво лишаться целой области обеспечения качества. Он велел мне вернуться к работе. Потом зачем-то завел длинную историю о том, что он лично был связан с одной из ветвей семейства Кеннеди, что у его родственников очень много денег и здание на углу Бродвея и Астор-плейс и что ему в общем-то нет необходимости работать. (Я еле удержался от вопроса, зачем он тогда вообще сюда приходит.)

Я все это проглотил, вернулся на свое место и начал просматривать новую пачку резюме, чтобы найти претендентов на две должности. Следующие полтора дня я обзванивал кадровые агентства и общался с кандидатами. В итоге я нашел трех, которые показались дельными. На мой взгляд, у каждого из них был неплохой потенциал. И, что важнее, мне

казалось, что они хорошо впишутся в нашу команду по личным качествам.

К моему ужасу, ни один из трех так к нам и не пришел. Вместо этого Эрик нанял Ноама и Дастина, практически сразу же, не дав ни мне, ни кому-то еще из команды провести с ними собеседование.

Ноам казался компетентным программистом. Эрик очень разволновался из-за этого, беседовал с ним два часа, и Ноам, по словам Эрика, «проявил значительное уважение к вышестоящему». Проблема была в том, что будучи хорошим программистом, Ноам был самым большим спорщиком, какого я только видел. Он сцеплялся со всеми: с сотрудниками отделов продаж и обработки данных, со своими коллегами по команде, со мной, с Аньес, с Питером; однажды в лифте он даже повздорил с президентом компании. (Не думаю, что он знал, с кем имеет дело, но, полагаю, это бы его не остановило.) Большинство его конфликтов завершалось одинаково: он рассказывал историю о том, как командовал танком в Израиле на Ливанском фронте и его каждый день обстреливали. Это был непременный финал любого его спора.

Истории с участием Ноама становились все более странными. Одна дама из отдела обработки данных сказала мне, что ее коллега однажды утром перед работой видела, как Ноам нюхал сиденье ее кресла. Кто-то из отдела продаж как-то сказал мне, что они так его боятся, что начали за глаза называть его «уродом из кунсткамеры». Слухи плодились, и все они были скверными.

Тем временем Ноам, кажется, проводил гораздо больше времени за спорами, чем за программированием, а работа стояла. К счастью, Пол и Ленни довольно скоро поняли, что не хотят, чтобы он ставил под угрозу успех большого проекта. Они смогли уговорить Аньес, чтобы она озадачила его поддержкой старых скриптов Ленни. Остальные члены команды быстро научились избегать Ноама.

К сожалению, Ноам плохо кончил. Одна из секретарей пожаловалась в отдел кадров на то, что он отпускал неприемлемые комментарии о ее религиозных взглядах. Она была из Свидетелей Иеговы, и Ноам затеял с ней большой спор о Новом Завете.

Эрик отозвал меня и сказал, что вышестоящие «нацисты» заставляют его уволить Ноама. Он сказал, что не хочет этого делать, понимая, как это огорчит меня и остальных членов команды. Похоже, ему казалось, мы боимся, что любого из нас могут уволить в любой момент. Он не понимал, что все мы до единого будем счастливы, если Ноам уйдет.

«Держись меня, – говорил Эрик. – Если ты докажешь мне свою верность, я не дам тебя в обиду».

Не знаю, что он имел в виду.

Тем временем Ленни, Дайана, Пол и Анна завершали последний этап большого проекта, и результаты были действительно хорошими. Теперь, когда Эрик больше не просил меня читать резюме, я мог вернуться к работе над проектом. Элен ушла, и я знал, что не смогу заменить ее как тестировщика. Но нужно было браться за любую необходимую работу, а такой работой как раз было тестирование. Поэтому я какое-то время занимался этим – к счастью, она оставила полный набор контрольных примеров, – а оставшееся время программировал. Было так здорово снова вернуться в команду! Когда мы работали вместе, казалось, что все мелочные проблемы, атаковавшие нас, оставались в другом мире. Особенно когда мы собирались в переговорной на восьмом этаже, где Эрик не мог нас достать.

Другой нанятый Эриком сотрудник, Дастин, положил конец этой временной идиллии. Все в офисе носили рубашки и галстуки. Это было правило, по-видимому, не применимое к Дастину, который приходил в джинсах, толстовке и мятой футболке. Но если бы только это! Настоящая проблема состояла в том, что и все остальные правила были, очевидно, также неприменимы к Дастину. Например, правило приходить на работу хотя бы раз или два в неделю.

От него очень неприятно пахло. И, к сожалению, все жаловались на него именно мне. Вся команда – Ленни, Пол, Дайана или Анна – по очереди приходила ко мне жаловаться на его запах. Кроме них приходили и Аньес, и еще пара сотрудников отдела продаж. Увы, когда я рассказал о жалобах Эрику, его это мало заинтересовало, и он уж точно не собирался беседовать об этом с Дастином.

Эрик требовал, чтобы мы передали часть большого проекта Дастину. Но мы вышли на финишную прямую, и делить работу было очень трудно. По сути, единственное, чем мы еще не занимались, была интерактивная справка, и мы возложили эту задачу на Дастина. Следующие четыре недели он над ней работал. Но на самом деле он не сделал ничего полезного. Выяснилось, что он пытался с помощью какой-то загадочной возможности API Win32 добавить справочную опцию в меню Windows (это такое маленькое меню, которое появляется, если щелкнуть в правом верхнем углу окна). Это было абсолютно бессмысленно – за пару минут можно было добавить опцию «Справка» в главное меню программы. Но Дастин очень хотел добавить ее именно в меню Windows, где нашим клиентам никогда не удалось бы ее найти и использовать. Я спросил об этом Эрика. Он ответил, что это не только хорошее решение, но президент компании лично его, Эрика, попросил добавить такую функцию.

Дастин так никогда и не завершил свой проект. В конце концов, он ушел в одну из новорожденных, растущих как грибы после дождя

интернет-компаний. После его ухода я просмотрел, чем он занимался. Он не сделал ничего стоящего. Мы выкинули написанный им код полностью.

Я все больше сомневался в том, что президент действительно заказал такую странную справочную систему, поэтому решил узнать об этом у Питера. Правда ли, что президент лично попросил добавить такую функцию? Питер понятия не имел, о чем я. Именно тогда мы оба поняли, что у нас серьезные проблемы. Эрик все это время лгал нам.

Но нам даже было некогда толком это обсудить, потому что Пол, Дайана, Ленни, Анна и я наконец-то завершали большой проект. Мы снова пригласили Питера, Аньес и сотрудников отделов продаж и обработки данных. На этот раз все прошло отлично. Неловких моментов и вопросов «разве не предполагалось, что система сможет *то-то и то-то*» не было. Все – и Аньес, и отдел обработки данных, и отдел продаж, и Питер, и вся команда – были довольны полученным результатом.

Все, кроме Эрика. После встречи он отвел меня в сторону и спросил, что я думаю про Дайану и Анну. Действительно ли они участвовали в проекте или всю работу сделали Пол и я ? Я сказал ему, что они были незаменимыми.

«Я поговорил с Питером, – сказал Эрик. – Он считает, что работу команды надо отдать в аутсорсинг. Не пойми меня неправильно, я считаю, что вы великолепные специалисты. Но Питер пытается убедить свое руководство и президента в том, что у команды серьезные проблемы. Мне очень неприятно говорить тебе это, но он считает, что вас всех нужно уволить».

Я был потрясен. Никогда я не слышал от Питера ничего подобного. Я не мог поверить в то, что меня собираются уволить. Эрик сказал, чтобы я не волновался. Он знал, что я «лоялен» к нему и всегда его поддерживал. У него все еще были знакомые в старом инвестиционном банке, готовые принять на работу всю команду целиком в любое время.

Это было уже последней каплей. У меня всегда были хорошие отношения с Питером, поэтому я пошел прямо к нему. Я сказал, что не могу поверить в то, что он хочет нас уволить и нанять внештатных специалистов. И начал приводить свои аргументы.

«Эндрю, я понятия не имею, о чем ты».

Оказывается, Питер только что говорил с Эриком, и тот сказал, что считает Дайану и Анну очень хорошими работниками. Но, сказал он Питеру, посмотрев на их работу в проекте, *я* решил, что их надо уволить. А кто такой Эрик, чтобы спорить с мнением главного члена команды?

Видимо, Эрик лгал всем. Мне, Питеру – да, пожалуй, всем, с кем он вообще говорил. И словно ему было мало одной лжи, он с самого начала вызвал целый ряд провалов: проект интранета, Элвис с его наркотиками, Элен и ее зарплата, печальные истории с Ноамом и Дастином. Более того, что касается той ставки, о которой он в самом начале своей работы у нас говорил, что с боем выбил ее, – никакой борьбы не было. Оказалось, что Питер, когда привел Эрика, уже убедил высшее руководство в том, что нам нужен еще один программист. Эрик знал об этом и лгал мне в лицо месяцами.

На следующий день Эрика уволили.

Всю неделю я чувствовал свою вину за то, что случилось с командой. Элен уже ушла, и я боялся, что Анна, Пол, Дайана и Ленни тоже пали духом. Я боялся, что они будут плохо ко мне относиться из-за того, что я недостаточно адекватно реагировал на действия Эрика. Я бы понял их, если бы они подали заявления об уходе. Всю неделю я прятался, потому что, как мне казалось, все думают, что это я их подставил. Мне позвонил Питер и попросил зайти к нему.

«Я поговорил с командой, – сказал он. – Нужен новый руководитель вместо Эрика, и каждый ответил, что это должен быть ты. Если хочешь – место начальника твое».

На этот раз я согласился.

25

Знакомьтесь: процесс разработки

Добро пожаловать за кулисы

Нед Робинсон

Это было в апреле 2001 года, небеса Нью-Йорка как раз сменили зимнюю серость на ясную весеннюю голубизну. Программист с двадцатилетним стажем, предыдущие пять лет я проработал в подразделении программного обеспечения компании RePlay Technologies, производителя музыкального софта. Дела у нас шли не так уж плохо для небольшого стартапа, но через пять почти безубыточных лет RePlay продали, и я оказался без работы. И вот я снова в финансовом районе Нью-Йорка в поисках работы.

Уже на первых собеседованиях я понял, что подход к разработке ПО в американских компаниях за прошедшие пять лет не изменился. Формализации процессов и качественному управлению проектами попрежнему уделяли очень мало внимания. В каждой компании повторялся один и тот же припев: работать 60–70 часов в неделю, титанический труд приведет проект к успеху (красный плащ супермена и зуммер выдают, как приступишь к работе), и да, будьте готовы к тому, что заказчик постоянно меняет требования. Так когда начинаем?

Проблема была в том, что меня уже не прельщала карьера разработчика-супермена. Я слишком долго им был и перегорел. Когда мне не было еще тридцати, и я зарабатывал себе имя в отрасли, я был готов к этому. В любых проектах, при любых обстоятельствах я долгими бессонными ночами строчил и строчил код, пытаясь удовлетворить неполным и плохо сформулированным требованиям, которые менялись уже на следующее утро, когда босс приходил в офис с новыми идеями, пере-

черкивающими 80% сделанного мною к тому моменту. Но я уже давно не ощущал адреналиновый всплеск, вручая начальнику работающую программу со всеми изменениями, о которых он сообщил только вчера вечером перед уходом. Я больше не хотел жить в мире постоянно расползающихся требований. Не хотел больше работать в команде невыспавшихся нервных программистов.

Размышляя обо все этом, я вошел в вестибюль здания 2 Всемирного торгового центра, направляясь на собеседование в Benchmark, небольшую программистскую компанию, производящую настольные инструменты оценки акций для управляющих активами и портфелями. Я вознесся на лифте на 78-й этаж и прошел в их офис.

Помещения были великолепны. Они занимали добрую половину 78-го и весь 77-й этаж, и везде, куда ни глянь, простирались ничем не заслоняемые самые невероятные ландшафты Нью-Йорка. В окне справа от стола секретаря в приемной высилась Статуя Свободы. По ощущениям, рабочая обстановка была чудо как хороша, и я надеялся, что работа окажется тоже на уровне. Записавшись в приемной, я стал ожидать менеджера по разработке.

Когда в приемную вышел Курт, я немало удивился (и обрадовался) его внешнему виду и поведению. Он был в черных джинсах, черной рубашке и черных армейских ботинках. Его прямые черные волосы с широкой фиолетовой прядью справа были собраны в хвост. Картину дополняла серьга в ухе. Внезапно во мне проснулась надежда на то, что эта работа может оказаться иной, чем я ее себе представлял.

Мы с Куртом спустились в его кабинет на 77-м этаже. Комната была завалена всевозможным компьютерным железом, жесткими дисками, материнскими платами, картами памяти и тому подобным. Наш разговор начался непринужденно, с моего рассказа о прежней работе. Выяснилось, что Курт тоже музыкант – в школе играл на скрипке, а сейчас басист в панк-группе. Мы поговорили о музыке и о моей работе над алгоритмами обработки музыки в RePlay Technologies. Обсудили славные времена программирования в бета-версиях компиляторов Майкрософт, поговорили о написании 16-битного кода в моделях с ограниченной памятью и о клиентах, которые обычно в последнюю минуту, когда ты уже почти все сделал, просят добавить несколько полей (во все окна и отчеты). Обсудили даже плюсы и минусы языка Форт и обратной бескобочной записи.

Потом он рассказал мне о команде разработчиков, которой руководил, и о модернизации процессов разработки, которую он начал. Вот это уже поинтереснее, подумал я. Он рассказал о том, что получил степень ба-

калавра в Карнеги-Меллоне[1], колыбели модели CMMI, и о своей работе по выводу компании на второй уровень соответствия по шкале CMMI. Завершив краткий рассказ о CMMI, он сказал то, на что я купился еще до того, как мне предложили работу: «Если тебе когда-либо придется работать больше 40 часов в неделю, значит, это я не справился как руководитель, а не ты провалил работу. Моя задача – нанять хорошего специалиста, и пусть он делает то, зачем его наняли. Для разработчиков ПО это означает внедрение на уровне отделов или всей компании проверенных временем процедур и технологий, четко определяющих все объемы и виды работ по проектам и позволяющих эффективно управлять изменениями».

Ого! Не должна ли земля разверзнуться и поглотить его за такое кощунство? Не сказал ли он только что, что изменениями объемов работ можно управлять? Что может быть прекраснее этих слов из уст руководителя? Неужели высшее руководство действительно пошло на это? Десятки вопросов проносились у меня в голове, но я уже чувствовал, что работа мне подходит. За следующие несколько дней я познакомился с командой разработчиков, прошел технический тест, оценку личностных качеств и тест на наркотики. Через неделю у меня была новая работа.

Я стал четвертым разработчиком в группе разработки ПО. Команда работала очень слаженно, и каждый участник был силен в чем-то своем, в чем на него полагалась вся остальная команда. Дмитрий был гуру битов информации и не боялся грязной работы в программировании. Питер любил преобразовывать бизнес-требования в отдельные задачи и создавать планы разработки программных продуктов. Я был не понаслышке знаком с внутренним устройством Windows, объектно-ориентированным проектированием и многоуровневой архитектурой. Наши сильные стороны делали нас ядром растущей команды, которой нравится работать вместе и писать код.

Команда также хорошо взаимодействовала с другими отделами системной разработки: определение бизнес-требований, обеспечение качества, управление проектами, производство и передовые коммерческие потребители. Общение было настолько теплым, что один из разработчиков женился на девушке-тестировщике из отдела обеспечения качества. Во имя сохранения гармонии в их семье мы делали все возможное, чтобы они не попали в один проект!

---

[1] Университет Карнеги-Меллона (англ. Carnegie Mellon University, CMU) – частный университет и исследовательский центр в Питтсбурге (Пенсильвания). – *Прим. перев.*

Во всей компании царила атмосфера, создававшая очень позитивные условия работы. Многие из примерно 200 сотрудников работали в компании уже по 5–10 лет. Все они старались способствовать росту компании, чувствуя себя ее частью. Такое отношение поддерживалось начиная от верхов компании и до рядовых работников. Вели учет круглых чисел по количеству пользователей, и достигнув очередного круглого числа (100 компьютеров или 10 новых пользователей), каждый сотрудник получал деревянную табличку и чек на 100 долларов. Те, кто работал в компании уже много лет, гордо восседали в отсеках, увешанных табличками от пола до потолка.

Дела у Benchmark шли хорошо. Компанию купила крупная фирма, занимавшаяся финансовой информацией, но Benchmark оставалась достаточно независимой, хотя и поползли слухи о том, что нас будут подгонять под формат новых хозяев. Но пока еще в Benchmark применялись собственные процедуры и методы.

Моим первым проектом стала программа, которая в реальном времени отслеживала критические процессы сбора данных на их конечной стадии. Программа должна была проверять поступление данных от поставщиков на серверы компании, а также автоматическое конвертирование, сохранение этих данных и их подготовку к передаче нашим потребителям. Она должна была также проверять работу защищенной системы отложенных запросов на котировки и точность выдаваемых котировок. Эта «сторожевая» программа должна была дополнить контрольную программу предварительной обработки данных, которую написал Дмитрий и которая круглосуточно работала на рабочей станции ИТ-отдела.

Начав записывать для себя варианты использования (use cases), я стал понимать, почему у Benchmark такие низкие показатели по разработке ПО в срок и в соответствии с техзаданием. Чтобы написать такую сторожевую программу, мне нужно было знать функции, предполагаемые для каждой системы, работа которой будет наблюдаться. Начав искать информацию по каждой системе, я обнаружил, что по многим из них документация отсутствовала или имелись только предварительные спецификации, которые совершенно не соответствовали конечному разработанному продукту. Например, файлы данных от поставщиков должны были конвертироваться в специальный формат, где к каждому ряду и в конце файла добавляется контрольная сумма. К сожалению, формат исходного файла данных, предоставляемого поставщиком, изменился – в структуру данных были добавлены поля. Поскольку мне нужно было проверять конкретные поля и вычислять контрольную сумму, я должен был убедиться, что считываются нужные поля. Усложняло задачу еще и то, что разработчик, написавший первоначальную

программу импорта данных, уже не работал в компании. Я попытался поговорить с Нэнси, главой отдела, занимавшегося передачей данных. Но у нее не было времени и терпения на то, чтобы заниматься сбором требований и обсуждать их со мной, и она сказала, чтобы я разбирался сам. В конце концов мне пришлось просто сделать следующее: открывать исходный код импортируемого файла и расшифровывать, что с ним происходит.

И на такие барьеры я натыкался постоянно, пока делал анализ вариантов использования. Единственное, что прошло гладко, – сопряжение с частью системы мониторинга, разработанной Дмитрием. Когда я спросил у него, как можно наладить интерфейс, он прислал мне ссылку на полные и точные спецификации API! С полностью документированными требованиями я смог закончить разработку сторожевой программы за несколько дней до поставленного срока.

Когда проект уже приближался к завершению, Курт назначил меня на еще один внутренний проект Benchmark. Одной из задач этой программы было отслеживать карьерные движения всех руководителей высшего уровня на Уолл Стрит, данные о компаниях, в которых они работают, их прежних и нынешних должностях, продвижениях по службе, зарплатах, премиях и так далее. Курт выдал мне черновик документа о требованиях, который ему предоставила команда Нэнси: краткое описание на одну страницу и пара рисунков окон. Совещание по запуску этого проекта было запланировано на следующую неделю.

Когда оно началось, всех участников представили друг другу. Нэнси привела на совещание шестерых членов своей команды, другие отделы прислали одного-двух человек. Один из помощников Нэнси рассказал о целях проекта и объеме работ. Когда они закончили, Курт спросил у Нэнси, когда ее команда сможет предоставить готовый документ по требованиям. По мнению Нэнси, того, что она уже дала, было вполне достаточно, чтобы понять все требования по проекту, и что они слишком заняты, чтобы писать или анализировать что-то еще. Поскольку свободных бизнес-аналитиков, готовых работать на нашем проекте, не было, разработчикам ПО нужно было самим все придумать. Мы с Куртом в замешательстве переглянулись. Не верилось, что это всерьез. Курт напомнил Нэнси о ее обещаниях, Нэнси ответила, что разработчики должны быть более гибкими и схватывать все на лету.

Курт стоял на своем, убеждая Нэнси, что без требований проект зависнет. В отличие моих бывших руководителей, Курт не собирался помалкивать. На этом менеджер проекта закрыл совещание, сказав, что оно будет продолжено после того, как вопрос требований будет улажен задействованными сторонами отдельно. Выходя из конференц-зала, я улыбался, думая о том, что Курт только что заработал немало очков.

К концу лета 2001 года я начал чувствовать себя в Benchmark как дома, словно я всегда работал именно в этой команде, хотя пробыл там меньше полугода. Однажды утром, в конце августа, придя на работу, я вышел из лифта и навстречу мне приветственным транспарантом развернулся невероятный вид на Статую Свободы в обрамлении ясного синего неба. В наушниках у меня играла «Рапсодия в блюзовых тонах» Гершвина. Это была одна из тех минут, которые помнишь до конца свой жизни: музыка, открывающийся вид, синее небо надо всем этим. Я вздохнул и легкой походкой направился к своему рабочему месту.

Меньше чем через две недели все это исчезнет навсегда, невероятной станет даже сама возможность случайного сочетания этого вида, этой музыки и цветов раннего утра – и единственным памятником безбрежности того переживания станут ощущения, хранящиеся в моей памяти.

Время приближалось к 8:30 утром 11 сентября 2001 года. Кто-то из нашей группы уже был в офисе, занимаясь своей обычной работой. Я проверял список вопросов, имевшихся у меня по проекту Нэнси. Было тихое утро дня, который, казалось, тоже будет тихим. Я прервался ненадолго, проверил, как идут дела у Барри Бондса[1] и не побил ли он еще рекорд Бэйба Рута[2] по скорости пробега до базы. Я уже собирался налить себе вторую чашку кофе, когда вдруг раздался взрыв. Мы с Питером подошли к окну, и почти сразу же мимо нас пронесся сгусток огня; вслед за ним летели дым и пепел. Наш угол обзора (окна выходили на восток, на Гудзон) не позволял нам увидеть, что вызвало взрыв и огонь. Мы подождали пару минут, когда же обслуживающий здание персонал сделает объявление, но стояла тишина. Элис (бизнес-аналитик), Питер, Меган и я после быстрого обсуждения решили начать спускаться по запасной пожарной лестнице, предполагая, что скоро поступят какие-нибудь инструкции. Выйдя на лестницу, мы увидели, что многим пришла та же мысль. Движение застопорилось, но не до конца, и все продолжали спокойно и организованно спускаться вниз. Кто-то нес пленки, устройства для резервного копирования, диски и другие личные вещи. Я нес ноутбук в сумке через плечо.

Чем больше людей выходили на пожарную лестницу, тем медленнее становилась процессия. Витавшая в воздухе тревога усиливалась от того, что мы до сих пор не знали, что произошло. Бомба террористов? Или несчастный случай, связанный с газом или электричеством? Ни один из тысяч людей, спускавшихся по той лестнице, не мог и поду-

---

[1] Барри Бондс – американский бейсболист, выступавший за команду Главной лиги бейсбола «San-Francisco Giants». – *Прим. перев.*

[2] Бэйб Рут – легендарный бейсболист, игрок Главной лиги бейсбола с 1914 по 1935 год. – *Прим. перев.*

мать, что причина – самолет, протаранивший Северную башню, соседку нашей. Не подозревали мы и того, что второй самолет уже мчится к Южной башне, чтобы ее атаковать.

Мы вчетвером держались вместе, пытаясь подбодрить друг друга. Через 45 минут мы достигли 50-го этажа. Из коридора донесся приглушенный звук объявления, и все вышли с лестницы, чтобы послушать. Отсутствие громкоговорителей на лестнице – скорее всего, просчет проектировщиков. Или нет?

Скрипучий мужской голос предлагал сохранять спокойствие, не сообщая о причине взрыва. Он просто успокаивал нас тем, что ситуация в Северной башне под контролем, и, вот уж где ирония, просил всех вернуться в свои офисы и ожидать дальнейших распоряжений.

Возникла дилемма. Мы все пытались оставаться организованными и следовать инструкциям. К сожалению, на подсознательном уровне эти инструкции всем казались бессмысленными. Что же делать? То, что нам подсказывала интуиция, – выбираться оттуда как можно скорее, или то, к чему призывало радио, – возвращаться к рабочим местам? Меган и Питер отошли, чтобы обсудить это самим, нам с Элис предстояло решать за себя.

Несколько раз передумав, мы в итоге решили, что нужно спуститься, выпить кофе и посмотреть, что будет дальше. Мы с Элис вернулись на пожарную лестницу, которая теперь была пустой, поскольку все еще решали, как поступить. Теперь мы могли бежать вниз. И мы бежали и бежали, пока вдруг не услышали сильнейший взрыв. Лестница заходила ходуном, чуть не на метр в каждую сторону. Я схватился за перила, подспудно ожидая, что они отвалятся от стены, ведь лестница превратилась в маятник. Почему-то вспомнилось, как Шелли Уинтерс в фильме «Приключение „Посейдона“» хватается за металлическую приставную лестницу, которая отходит от стены, и она падает в воду. К моему большому облегчению лестница в нашем здании выдержала и качка прекратилась. Я заметил, что мы сейчас были на 43-м этаже. Удивительно, но лестница вообще не пострадала. Свет продолжал гореть, лестничные коробки и перила были целыми, и нигде не лежало ни кусочка отвалившихся краски или гипса. Элис заговорила первой, предложив продолжить спуск. Я согласился, и мы бежали, пока другие не вышли вновь на лестницу и скорость не упала почти до нуля.

Нам понадобилось около 45 минут, чтобы покинуть здание. В торговом павильоне Всемирного торгового центра, когда мы зашли, был организован живой коридор из пожарных и полицейских, по которому нас проводили наверх к зданию 5 (или 7?). Они достойно делали свою работу, помогая нам выбраться. После я каждый день молился о том, чтобы они тоже смогли покинуть здание прежде, чем оно рухнуло.

Всего наша компания из 200 сотрудников не досчиталась в тот день четверых. Мы все чувствовали себя одной большой семьей. Прекрасного офиса, который достался нам нелегко и был символом нашего успеха и единства, больше не было. Мы стали бездомными из-за незабываемой трагедии. И мы потеряли четырех товарищей. Но как хорошая команда мы сплотились и безустанно трудились, чтобы снова поставить компанию на ноги, чтобы поддерживать друг друга духовно, эмоционально и во всем остальном.

Было много разных историй о том, как люди смогли выбраться из Южной башни, и я хотел бы рассказать о том, как боролись за жизнь некоторые из членов нашей команды. Хочу сказать также, что самолет, врезавшись в Южную башню, угодил прямо в наш конференц-зал на 78-м этаже. Это был просто идеальный конференц-зал. Вся наружная стена была стеклянной, и оттуда открывался невероятный вид на южную оконечность Манхэттена. Это было настолько хорошее место, что наша головная компания часто проводила в этом зале важные презентации. В то утро двое из этой компании пришли к нам пораньше, чтобы подготовить презентацию. К сожалению, они погибли в результате атаки.

Апарна была одним из членов нашей команды по обеспечению качества. 11 сентября она была на девятом месяце беременности. Услышав первый взрыв, она решила остаться на своем рабочем месте. Когда самолет врезался в Южную башню, перекрытия на 77-м этаже рухнули, наполнив все вокруг пылью и обломками. Беременность помешала Апарне залезть под стол. Ей оставалось стоять под градом обломков. Когда пыль начала оседать, она позвала на помощь. Прибежали Рик, ее начальник, и Тони, менеджер по работе с данными. Они провели Апарну к пожарной лестнице и втроем спустились с 78-го этажа.

Когда они выбрались, Апарну немедленно отвезли на скорой в больницу, чтобы убедиться, что с ней и ребенком все в порядке. Все оказалось хорошо, и через месяц Апарна родила здорового мальчика. Рик был весь в пыли, и врачи дали ему кислородную маску. Он сидел в ней на бордюре у Черч-стрит. Внезапно полиция объявила, чтобы все убегали как можно быстрее. Здание начало рушиться. Срывая на ходу кислородную маску, Рик бежал наперегонки со смертью. Он бежал по Джон-стрит, но ему некуда было деться – большинство контор к тому времени уже позакрывались. Вдруг он увидел открытый бар, схватился за ручку двери и ввалился внутрь. Когда он закрывал дверь и был уже в безопасности, мимо прокатилась волна останков Южной башни.

Берни был ИТ-специалистом. Придя в то утро, он сразу отправился в конференц-зал на 78-м этаже, чтобы помочь гостям подготовиться к презентации, проверив работу аудио- и видеосистем. Он привел все

в рабочую готовность и вернулся в свой кабинет на 77-м этаже, случайно оставив в конференц-зале свой портфель; в это время первый самолет врезался в Северную башню. Какое-то время он пытался связаться с кем-нибудь из соседей и уже было собрался уйти, когда заметил, что забыл портфель. Он позвонил Виолетте, чей офис был рядом с конференц-залом.

«Виолетта, это Берни, я потерял портфель. Посмотри, пожалуйста, не оставил ли я его в конференц-зале? И, кстати, почему ты все еще там?»

«Да, Берни, портфель тут. Я его вижу. Переключу телефон на автоответчик и буду выбираться», – ответила Виолетта.

«Хорошо, спасибо, – сказал Берни. – Я сейчас позвоню маме, скажу что со мной все в порядке, а потом заберу портфель и тоже уйду. Спасибо».

Берни быстро поговорил с матерью, и в тот момент, когда он прощался и клал трубку, Виолетта, которая еще оставалась на рабочем месте, увидела, как тень закрыла окно конференц-зала и второй захваченный самолет врезался в окно. Виолетта, замерев в своем кресле, в ужасе смотрела, как всего в паре метров от ее лица проносится крыло самолета. К счастью, она смогла заставить себя очнуться и побежала к запасному выходу.

Берни тоже направлялся к запасному выходу. Первый выход, к которому он подошел, был заклинен. Он пытался открыть дверь, но что-то мешало с той стороны. Он побежал через холл к другому выходу, на этот раз дверь открылась. Однако площадка была залита какой-то жидкостью, текущей с потолка. Почувствовав запах бензина, Берни решил, что пробило трубу системы отопления. Он не знал, что на него льется авиационное топливо.

Еще один сотрудник, Энтони, выбежал на лестницу с фонариком в руках. Они с Берни побежали по лестнице вниз. Коснувшись перил, Берни заметил, что они были липкими. Он решил, что на перила, как и на него, пролился бензин. Они пробежали вниз около 30 этажей, когда внезапно погас свет. В полной темноте Энтони включил фонарик, и следующие 10 этажей они прошли несколько медленнее, пока не добрались до освещенного места. Здесь Берни увидел свои руки. Они были в крови. Жидкость на перилах верхних этажей не была бензином. Это была кровь погибших.

Берни и Энтони спустились в вестибюль Всемирного торгового центра и выбежали на улицу. К Берни, который был весь в бензине и в крови, сразу же подошел спасатель и помог ему снять рубашку. В это время полиция объявила: «Бегите!», и Берни побежал, но ему повезло меньше, чем Рику. Не найдя убежища, он просто продолжал бежать, даже когда

облако обломков догнало и накрыло его, как оно накрыло весь Нижний Манхэттен.

В тот же день я созвонился со всеми членами команды разработки и убедился, что все целы, в безопасности и о них есть кому позаботиться. Я также искал добровольцев, которые поехали бы в наш небольшой офис в Филадельфии, чтобы там восстановить серверы данных, которые были в Нью-Йорке, и наши многочисленные процедуры для круглосуточной поставки данных клиентам.

В полпервого ночи, когда я собирался ложиться, зазвонил телефон.

«Привет, это Нед? – спросил звонивший. – Я друг мужа Барбары. Барбара работает с вами в Benchmark, правильно?»

«Да, это я», – ответил я.

«Ее никто не видел и не слышал с сегодняшнего утра. Ее муж спрашивает, может быть, вы ее видели?»

«Нет, – беспомощно прошептал я. – Я видел ее у кофе-машины рано утром. Но наши рабочие места в разных концах здания, и после я ее не видел».

«Хорошо, спасибо, – сказал друг. – Ее муж просто хочет обзвонить всех, кого только можно».

Я повесил трубку. Барбару так и не нашли, и в конечном итоге она стала одной из четырех сотрудников Benchmark, которых сочли погибшими в тот день.

На следующий день 15 сотрудников Benchmark отправились в Филадельфию. Там был офис небольшой группы, которая программировала и поддерживала интерфейсное приложение. У них был один конференц-зал, который собирались переделать в центр данных, и довольно тесные отсеки, занимаемые разработчиками. Очень скоро офис стал похож на университетский городок в последнюю неделю перед экзаменами.

Нам нужно было привести ПО в полную рабочую готовность к открытию биржи в следующий понедельник, чтобы можно было отправлять данные всем клиентам. То есть у нас было меньше недели на то, чтобы установить и настроить серверы и каналы отправки данных, а также организовать круглосуточную техподдержку.

Курт и Дмитрий представляли группу программного обеспечения. Всю следующую неделю они без устали трудились над тем, чтобы все разработанные нами программы были загружены на серверы и правильно настроены. Остальная часть команды поддерживала с ними постоянную связь из Нью-Йорка, помогая проводить нагрузочные испытания каналов данных по мере того, как их подключали, и проверять возможность применения программы клиентами. Мы редко спали, почти не

возвращаясь в отель. Было ощущение, что мы работаем в центре НАСА во время полета на Луну! В каждом углу, не занятом сервером данных или разработчиком с ноутбуком, до потолка громоздились коробки из-под пиццы.

К тому времени как в следующий понедельник открылась биржа, наше ПО было готово к работе, хотя и с некоторыми заплатками. Однако наши клиенты были рады, что работал хотя бы один из поставщиков данных.

Дмитрий остался в Филадельфии еще на несколько дней, чтобы провести переучет наших резервных серверов исходного кода. Необходимо было убедиться, что все рабочее ПО может быть скомпилировано и запущено согласно спецификациям. Дмитрий позвонил мне домой с отчаянием в голосе. Он сказал, что резервное копирование сервера SourceSafe, на котором хранилась вся документация компании, выполнялось каждую ночь, но те серверы, где хранился весь исходный код, никогда не были включены в систему резервного копирования. Ничего себе! Никто не позаботился проверить, полностью ли выполняется копирование серверов. Пропали все программы, написанные Дмитрием в Benchmarke за последние семь лет.

Следующие несколько недель мы только и пытались понять, какой исходный код у нас есть, а какого нет. Мы проверяли свои домашние компьютеры и все имевшиеся ноутбуки. К радости Дмитрия, его последняя крупная работа была сохранена в рамках одного из наших проектов по усовершенствованию процедур. Мы организовали технические экспертизы, проводившиеся сотрудниками в Нью-Йорке и Филадельфии, и недавно направили группе в Филадельфии большой проект, над которым работал Дмитрий. Исходный код сохранился на компьютере руководителя отдела разработки в Филадельфии.

В итоге высшее руководство решило, что, поскольку утерян такой большой объем исходного кода, восстанавливать будем только то рабочее программное обеспечение, которому срочно требовалось обновление. Это позволило группе заняться новыми разработками. Остался только вопрос, где мы будем работать.

Через несколько недель после 11 сентября вся компания собралась на встречу в клубе Union в Нью-Йорке. Это было наше первое собрание после теракта. В какой-то мере атмосфера клуба Union подчеркивала значительность этой встречи. Одетые в выходные костюмы, мы вместе почтили память наших погибших коллег. Одним из них был Джордж, специалист по продажам, много лет проработавший в компании. Его коллега из отдела продаж рассказал мне, что в то утро сидел у зубного, и когда по телевизору начались репортажи с места событий, почувствовал полное отчаяние. Он очень хотел быть в офисе вместе со своей ко-

мандой. Эти несколько недель он испытывал вину за то, что его там не было. Ему казалось, что тогда он мог бы спасти своего друга. Так думали многие: те, кто выходил в это время из метро, кто ехал в машине на работу и видел, как в нас врезались самолеты, те, кто жил неподалеку, по дороге на Бэттери-Парк. Это кажется нелогичным, но людям необходима хоть какая-то связь.

На встрече в клубе Union Нэнси заговорила со мной и Куртом о возможности отправить разработчиков в Филадельфию, пока не будет найдено новое помещение в Нью-Йорке. Она не верила, что разработчики могут эффективно работать дома и что им необходимо постоянное непосредственное руководство. Курт рассказал ей, что офис в Филадельфии уже переполнен и что он не может заставлять разработчиков неизвестно сколько времени жить в отеле. Он не сдавался, и Нэнси ушла обиженная и сердитая.

Прошло целых шесть месяцев, прежде чем все мы смогли переехать в новый офис. До этого все работали дома, делая это очень эффективно и постоянно общаясь с остальными группами компании. Мы общались по мессенджеру, созванивались, иногда встречались в нью-йоркских ресторанах вроде America или Zen Palate на Юнион-сквер, делая все необходимое для того, чтобы работать эффективно. Я даже провел три часа в Starbucks с одной из помощниц Нэнси, пытаясь выяснить требования по зависшему проекту. И узнал, что как раз она может это сделать. Нэнси разрешила ей помочь нам с требованиями, если это не будет мешать другой работе. К тому времени как мы переехали в новый офис, требования были подготовлены, проверены и утверждены, а половина кода уже написана.

В конце февраля 2002 года мы переехали. Мы заняли весь пятый этаж бывшей интернет-компании в Вест-Вилледж. Там была комната для настольного футбола, комната для занятий йогой, множество непрямых углов и стен нестандартной геометрии. Оказалось, что в одном из баров в Филадельфии был настольный футбол, и некоторые тамошние сотрудники стали в нем чуть ли не профессионалами. Комнату для занятий йогой переоборудовали под переговорную, но стол для футбола остался!

Нас ждала еще одна перемена. Курт решил вернуться в университет Карнеги-Меллон, чтобы продолжить заниматься CMMI и разработкой технологических процессов. Он уволился, и его место занял я. Кроме того, в команду взяли еще двух разработчиков и консультанта. Команда менялась на глазах. Но то, что мы пережили вместе за последние полгода, неразрывно связало нас невидимыми нитями. То, как мы относились друг к другу, как работали вместе, как смотрели на задачу

и без слов знали, какое решение приходит в голову коллеге, – нет, никто об этом не говорил, но все это чувствовали и гордились этим.

Одной из привилегий в нашем отделе, которую я учредил сразу же, была возможность для каждого разработчика один день в неделю работать дома. Мы уже доказали, что можем дома работать столь же эффективно, как и в офисе. Единственной сложностью было то, что нам приходилось делать это тайком, потому что другие начальники были против такой политики (угадайте, кто).

Команда по-прежнему много работала. Теперь, когда у нас появлялось свободное время, мы создавали прочную базу исходного кода. Каждый разработчик отвечал за написание кода, допускающего многократное использование, и по каждому своему проекту должен был создать хотя бы один многократно используемый компонент, например внутреннюю библиотеку DLL или виджет пользовательского интерфейса. На еженедельных собраниях отдела один из разработчиков представлял свою работу остальной группе. Приводил фрагменты кода, применяющие разработанный им элемент, и показывал всей группе свой код, написанный для этого объекта. Вскоре должна была подойти очередь презентации Дмитрия, старшего разработчика группы.

Дмитрий отказывался. Он пришел ко мне и сказал, что не готов выступать перед группой. У него был сильный русский акцент, ему казалось, что он не сможет нормально рассказать о своей работе и только сорвет собрание. Я ответил: «Дмитрий, ты постоянно встречаешься с одним-двумя сотрудниками, показываешь им, как писать код, или помогаешь отладить программу. Все знают, что ты прекрасный специалист. Не волнуйся, здесь все свои!» Мы еще поговорили, и я сказал, что, если он совсем не хочет делать презентацию, то может пропустить свою очередь. Он подумал немного и все-таки согласился попробовать.

Наступил день его презентации. Дмитрий, не успев распечатать раздаточный материал, опаздывал на собрание. Поисковая партия обнаружила его у копира сражающимся с подборочной машиной. Ему помогли устранить замятие бумаги и сделать недостающие копии. Вернулись все вместе, и Дмитрий начал презентацию. Он написал библиотеку DLL, позволявшую стандартизировать формат журналов ошибок, направляя вывод в текстовый файл либо в Windows Event Logger или в оба места. Это было очень полезно для разработчиков, они бы с радостью задействовали новую функцию. Когда он начал писать пример кода привязки к DLL на доске, было видно, что он очень волнуется. Он говорил, не оборачиваясь к аудитории, а написанное им было трудно разобрать. Он весь вспотел, с горем пополам закончил презентацию, сел и спросил, есть ли вопросы. Не помню, кто задал первый вопрос, но на лице у Дмитрия появилось выражение такого отчаяния, что все про-

сто рассмеялись. Даже сам Дмитрий. И этого хватило, чтобы снять напряжение. Собрание из формальной презентации превратилось в обычную встречу, где Дмитрий показывал остальным, как делать то, в чем он разбирался досконально. Встреча все продолжалась, и мы даже не заметили, как вышло наше время и уже следующая группа стучала в дверь, чтобы мы освободили переговорную.

Меган так же боялась публичного выступления, как и Дмитрий. Она была одним из тестировщиков. Она обладала упорством и въедливостью, позволявшими выявлять даже самые незаметные ошибки. Когда она шла в отдел разработки, программисты начинали прятаться. К сожалению, на обсуждениях проверок она чаще молчала. Однажды я сидел рядом с ней на собрании, посвященном проверке требований. У каждого была своя копия документа. Взглянув на копию Меган, я был поражен: она пометила красным чуть не каждый абзац! Она действительно все внимательно вычитала. Однако на собрании так ничего и не сказала. Я подумал, что у нее были на то причины, и промолчал.

Позже в тот же день я подошел к ее столу.

«Меган, есть минутка?» – спросил я.

«Конечно. Что такое?»

«Просто любопытно, почему ты не поделилась своими комментариями сегодня утром на собрании».

Вопрос ей явно не понравился, и она не ответила. Не будучи ее руководителем, я решил не давить на нее. Но позже поговорил с ее начальником, и мы вместе с ним начали мягко подбадривать ее.

Несколько месяцев спустя я вновь вместе с Меган был на собрании по обсуждению требований. Мы быстро проходили по документу, когда Меган вдруг подняла руку и сказала: «Извините, я нашла ошибки на предыдущей странице. Кому-нибудь это интересно? Не уверена, что это важно. Извините, что прервала». Мы все вернулись на предыдущую страницу, и Меган показала нам две найденные ошибки. Они оказались мелкими и некритичными, но все были рады, что их обнаружили. «Отличная работа, Меган», – сказал я. Она начала извиняться за то, что нашла их и отвлекла всех, но я ее перебил: «Нет, ты абсолютно права, что сообщила о них. Запомни, для нас гораздо лучше найти любые ошибки сейчас, чем на этапе разработки или тестирования. Спасибо». Со временем Меган заслужила всеобщее уважение как самый дотошный ревизор документации в отделе обеспечения качества.

Для меня как руководителя группы важно было понять, что у каждого свои сильные стороны и свои желания при написании кода. Мэри умела замечательно создавать пользовательские интерфейсы. Питер был главным специалистом по процессам. Мы вместе с ним изучили

все хитрости унифицированного процесса RUPRational (RUP) и создали так называемый унифицированный процесс Benchmark. Проект занял несколько месяцев, в результате была разработана подробная последовательность действий, позволявшая любому разработчику быть уверенным, что его проект соответствует всем процедурам цикла разработки ПО и что имеются все необходимые документы и элементы. Этот документ стал неоценимым активом отдела и ключом к разработке качественного ПО. Мы внедрили эту процедуру и следовали ей, при этом подчеркивая и делая ставку на сильные стороны каждого из нас, а не на его недостатки, в результате чего следующие три года команда разработки ПО безупречно и своевременно выполняла поставленные задачи в рамках бюджета и со всеми требуемыми характеристиками. В нашем отделе захотели работать все. Мне пришлось вести список тех, кто хотел перевестись в мой отдел.

Однажды ко мне пришел Берни, мой руководитель: «Есть просьба, Нед».

«Конечно, – ответил я. – Говори».

«В ИТ-отделе есть парень, который писал для нас небольшие утилиты на Visual Basic. И очень неплохо это делал. Я только что общался с ним по итогам года и выяснил, что ему скучно работать в ИТ. Мне кажется, если я не дам ему шанс разрабатывать что-то еще, он от нас уйдет».

«Берни, – сказал я. – Понимаешь, у меня есть еще пять официальных запросов о переводе в команду разработки. Ты готов расширить мой отдел еще на шесть человек?»

«Конечно, нет, – сказал Берни. – Но я не хочу терять Дага. Поговори с ним, и если он подходит вашей группе, я сам займусь его переводом».

«А можно, я возьму Дага и еще одного человека из списка? Знаешь, все это младшие программисты, и нужно время, чтобы научить их. Мы можем на несколько месяцев подключить их к опытному разработчику, и он их натаскает. Плюс, если выяснится, что хороших разработчиков из них не выйдет, мы поймем это довольно скоро – до того, как станет слишком поздно».

Берни сказал: «Пока не могу ничего обещать. Посмотрим, как пойдет с Дагом. Если получится, я подумаю, что можно сделать, чтобы открыть еще одну вакансию. Хорошо?»

«Согласен, – сказал я. – Поговорю с ним и сообщу тебе о результатах».

До конца месяца в группе появились два младших разработчика, Даг и Генри.

Хотя мы успешно внедряли процесс, следуя им и убедительно доказывая, что они работают и что более тщательное предварительное плани-

рование гарантирует лучшие результаты, несколько старших руководителей продолжали бороться с процессом при любом удобном случае. Мэри работала в группе подготовки отчетов для Нэнси, которая каким-то образом умудрилась избежать внедренного процесса. Одним из результатов в рамках отчета был настраиваемый расчет. У нас не было никаких спецификаций на то, как он должен работать, поскольку Нэнси и ее подчиненные просто сказали Мэри обращаться с вопросами к ним. Но расчет был действительно сложным, и даже сами коммерческие пользователи не вполне его понимали. Мэри только что закончила в третий раз переделывать отчет, а конечные пользователи все еще искали ошибки в расчете. Не зная, как достичь консенсуса по этому расчету, Мэри пришла в мой кабинет.

Я созвал на совещание все заинтересованные стороны, включая Нэнси, ее сотрудников, Мэри и меня. Всего присутствовало около 12 человек. Я сказал, что мы написали код для этого расчета в соответствии с той информацией, которую нам дали, но, похоже, разработчики так и не получили все возможные сценарии, которые могли повлиять на расчет. В результате от расчета ждали, что при разных условиях будут выполняться разные задачи. Потом мы попросили будущего пользователя схематично изобразить, как он представляет себе этот расчет.

Через 20 минут схема заняла всю доску. В этом поучаствовали все, и в итоге группа пришла к консенсусу. Да, все готовы были подписать схему в таком виде. Но тут еще один участник спросил об условии, которое не было отражено на доске. Все вздрогнули, поняв, что забыли важное условие, влиявшее на расчет в 15% случаев. Мы добавили его к схеме на доске. И снова все согласились с измененной версией и были готовы ее подписать. Тогда ключевой игрок Нэнси, которая была старшим из руководителей на этой встрече и постоянно сопротивлялась внедрению процесса, привела пользовательский рабочий поток, который снова полностью менял порядок расчета в 50% случаев! Все остальные это пропустили. Мы добавили его к схеме на доске, решив, что теперь наш расчет наконец-то приобрел совершенный вид.

У Мэри голова шла кругом, но она постаралась еще раз окончательно пройтись по схеме вместе со всеми. Когда она закончила, Нэнси посмотрела на нее, поднялась и сказала всем: «Кажется, теперь я поняла, почему процесс так важен. Мы должны были сделать это в самом начале проекта, и я приношу группе свои извинения за то, что игнорировала ваши просьбы об этом. Давайте я заберу эту схему к нам в отдел на пару дней, назначу бизнес-аналитика, и мы составим документ, отражающий наши точные требования. Пока мы не сделаем это, не тратьте время на программирование».

Я чуть со стула не упал! Это была большая победа. Курт и Нэнси воевали по поводу необходимости процесса с незапамятных времен, и я знал, что сегодняшнее заявление Нэнси стало большой моральной победой Курта, который к этому времени уже ушел из компании. Приобрести такого союзника, как Нэнси, в деле формализации процесса работы было большой победой и для всей команды разработчиков ПО.

Нэнси встретила меня у выхода из переговорной. Пока мы шли на свои рабочие места, она сказала мне: «Нед, я узнала, что ты разрешил своей группе раз в неделю работать дома. Даже по пятницам». О нет, подумал я. Только не это! Я начал что-то плести, но толком не знал, что сказать.

«Не волнуйся, – продолжила Нэнси. – Хотела только попросить, чтобы вы не слишком об этом распространялись в других отделах. Вижу, что у вас это вполне работает, но в моем отделе такое никогда не пройдет».

«Конечно, – сказал я. – Не вопрос. До встречи».

Я ушел к себе в кабинет, сел за стол и широко улыбнулся.

26

Преодолевая препятствия

Интервью со Скоттом У. Амблером

Скотт У. Амблер был среди тех, кто возглавил две революции в компьютерном мире: переход к объектно-ориентированному анализу и проектированию в 1990-х и внедрение динамичных (Agile) методов разработки десятилетием позже. Его деятельность связана с внедрением изменений в масштабе целых предприятий, а такие подвиги предполагают постоянное преодоление препятствий. Нам хотелось узнать, как ему удавалось добиваться успеха в таких изменениях, особенно когда окружающие не верили в их действенность.

Дженни: **Нас интересует ваш опыт и то, как вы пришли к пониманию механизмов работы в команде.**

Скотт: В настоящее время я руковожу Agile-разработкой в IBM Software Group. Моя работа в основном заключается в том, чтобы помогать заказчикам работать более гибко и эффективно. Я работаю и в рамках самой IBM. Во многом это именно обучение более эффективной совместной работе, часто в командах. Я написал несколько книг и провел немалую работу в сфере продвижения Agile-методологии. Я автор таких методик, как Agile Modeling (гибкое моделирование), Agile Data Method (метод гибких данных) и Enterprise Unified Process (корпоративный унифицированный процесс разработки).

Что касается моего опыта работы в командах, думаю, я много за него заплатил. Программное обеспечение большей частью разрабатывается в командах. В результате вы все равно учитесь работать вместе, и тут можно пойти по простому или по сложному пути. Полагаю, мой путь был сложным.

Дженни: **Убедить компании внедрить те методы и практики, о которых вы говорите, значит убедить многих изменить сам подход и отношение к своей работе. Как вам удается осуществлять такие крупные изменения?**

Скотт: Тут нужно просто браться и делать. Разработка ПО – это сложная, тяжелая работа, кроме того, это работа с людьми. Мне кажется, первое препятствие, которое необходимо преодолеть, – это ты сам. Нужно признать, что в одиночку тебе не справиться. Понадобится помощь и придется учиться у других.

Если говорить о препятствиях, очень важно суметь понять, что команда не работает, как нужно; это не всегда очевидно. Если люди кричат друг на друга, *возможно*, что-то у них не в порядке. Но однажды я работал в команде, о которой посторонний наблюдатель непременно сказал бы, что в ней все готовы поубивать друг друга. На самом деле, неизвестно почему, но лучше всего мы взаимодействовали в спорах. Там мы приходили к согласию, хотя при этом могли и здорово накричать друг на друга. Тем не менее мы очень эффективно работали вместе. В том числе и потому, что научились уважать друг друга. И хотя вслух это не говорилось, все мы знали, что для нас процесс разработки заключается в обсуждении и спорах. Неизменно повторялась ситуация, когда в начале ни у кого не было правильного мнения, но в споре мы находили наилучшее решение вопроса, над которым работали.

Я веду к тому, что не всегда можно определить, плохо ли команда функционирует. Например, в том случае человек со стороны подумал бы, что у нас не все в порядке, но на самом деле мы были феноменально

эффективны. Если люди кричат в негативном ключе, – наверняка, есть проблема. Но ведь бывает и так, что в команде все стараются, но при этом результаты не очень высоки. Это может быть неочевидным и обнаружиться не сразу, и это скользкий путь, когда все упорно и долго работают, но общие достижения невелики. Простая аналогия: одни днем копают яму, а другие ночью ее закапывают. И те и другие работают очень упорно, но ничего хорошего из этого не выходит.

Именно это может стать причиной неэффективной работы команды, при этом не всегда можно понять, что происходит, особенно если команда большая и распределенная. Очень трудно отслеживать такие вещи в командах, не представляющих собой единое целое.

Эндрю: **То есть препятствием может стать сам способ общения команды, причем он мешает не только делать работу, но и оценивать эффективность команды. Возможно, очень серьезной проблемой могут стать личные конфликты. Если в команде хорошие технические специалисты, но по какой-то причине они не объединяются, может пострадать дело. Полагаю, глубоким интровертам и тем, кто старается избегать конфликтов, было бы сложно попасть в ту вашу команду, да? Даже если с технической точки зрения они были бы идеальны, мне кажется, у них не было бы шансов. Но иногда проблемы личной несовместимости вскрываются только по ходу дела. Что вы делаете в таких случаях?**

Скотт: То есть в команде, скажем, есть интроверт, который попадает в итоге под перекрестный огонь, или что-то в этом роде? Кто-то должен это заметить. Интроверту нужно встать и сказать: «Знаете, мне все это не нравится». Но именно ему непросто это сделать. Предположим, он молчит. Тогда это должен заметить кто-то другой, что, конечно, тоже непросто. В той нашей команде это заметил бы руководитель.

Эндрю: **Вы считаете, что руководитель должен замечать такие вещи?**

Скотт: Хотелось бы надеяться. Хороший руководитель или лидер группы должен заботиться о каждом в своей команде, что, конечно, в некоторых ситуациях становится сложной задачей. Любой, кто возглавляет команду, должен быть внимателен к своим сотрудникам и задаваться вопросами: «Как дела у этого человека? Как он себя чувствует в команде? Чем я могу ему помочь? Какие у него проблемы?»

Нужны специальные методики, помогающие замечать такие вещи, поскольку отчеты о ходе работ здесь не всегда помогают. Интроверт наверняка не напишет в отчете: «Мне очень не нравится работать с этими людьми. Я не справляюсь с задачей и не могу ладить с этими людьми». Это должен заметить руководитель.

Допустим, кто-то обратил внимание, что у человека проблема и он плохо ладит с другими. Думаю, он должен поднять этот вопрос в группе или прямо сказать экспрессивным экстравертам: «Эй, это вам нравится кричать, а Салли чувствует себя неуютно. Может, попробуем и ее включить в обсуждение? Пожалуйста, полегче, научитесь работать и с ней тоже». Каждому в команде придется найти свой подход к Салли, возможно, и ей понадобится пойти им навстречу. Мы никогда не упускали возможности научиться чему-то, что для меня признак высокого профессионализма. Нужно постоянно стремиться стать лучше – как в работе, так и в поведении. Особенно важно учиться сотрудничеству, работе с людьми и в команде.

Для технических специалистов это может стать сложной задачей. В нашей области очень ценятся технические навыки, и многих очень поощряют за развитие именно в этой области, но мы часто недооцениваем так называемые человеческие качества, умения. В результате мы уделяем им мало внимания. А они очень важны, по крайней мере, в разработке ПО. Люди – главный фактор успеха команды разработчиков. Успешность вашего взаимодействия друг с другом гораздо важнее новой технологии, с которой вы сегодня работаете, как бы крута она ни была. Но обычно этот фактор недооценивают.

Эндрю: **Думаю, многие будут удивлены, услышав об этом; вы столько говорили о вопросах архитектуры, разработки и о методиках – объектно-ориентированной разработки, моделирования ПО и унифицированного языка моделирования, Agile-методологии разработки, планирования и разработки в целом. Но когда речь зашла о самых больших проблемах, с которыми могут столкнуться команды, вы говорите о проблемах человеческих взаимоотношений. Для вас не было неожиданностью осознать это? Или вы понимали это с самого начала?**

Скотт: Да, я был удивлен, впрочем, обнаружил я это довольно скоро, на основе опыта и наблюдений. Я действительно рано понял, что в нашей работе важны не только технологии. Не могу сказать, что я сразу пришел к пониманию важности человеческого фактора. Думаю, первым шагом было осознать, что нужно думать не только о технологиях, но уделять внимание и методикам. В магистратуре я работал над темой «Коллективная работа с компьютером» (Computer Supported Collaborative Work, CSCW), то есть над ПО коллективного пользования (groupware). Думаю, именно тогда я по-настоящему понял важность человеческого фактора в разработке ПО, потому что именно эту сферу охватывает CSCW. То есть что технологии помогают обеспечить более эффективное сотрудничество. Тогда я и начал задумываться над этим аспектом разработки.

Дженни: **Немалую часть своей карьеры вы посвятили лекциям и исследованиям по методикам. Бывало ли, что именно применяемые методики помогали преодолевать препятствия в коллективной работе, о которых вы говорите? И наоборот, могут ли сами методики стать таким препятствием?**

Скотт: В методиках неодиночной разработки, таких как парное программирование – или, в терминах Agile-моделирования, «совместное моделирование» – понимаешь, что работать вместе с другими необходимо, и постоянно работаешь в сотрудничестве. Многие люди, применяющие Agile-методы разработки, практикуют так называемые беспорядочные пары, где работа не просто ведется в парах, но и постоянно меняется состав этих пар. При этом не только повышается качество работы, но и появляется возможность многому научиться у других. Если состав пар регулярно меняется, эта возможность превращается в необходимость, и вся команда поневоле осваивает навыки коммуникации и взаимодействия, чтобы «выжить» в такой среде.

Но, как вы правильно заметили, этот метод может сам по себе стать непростой задачей, не каждый может работать таким образом. Это одна из сложностей, с которыми сталкиваются люди в Agile-разработке. В определенном смысле, такие навыки, о необходимости улучшения которых мы сейчас говорим, сами могут стать для некоторых непреодолимым барьером. Есть люди, которые действительно хотят концентрироваться на технологиях и работать самостоятельно. И это нормально. По моему мнению, организации и команды должны признать: все люди разные, есть те, кто не желает работать в команде. И есть такая работа, которую нужно выполнять самостоятельно. Мне кажется, это важное наблюдение.

Эндрю: **В ходе написания этой книги, очень многие рассказывали нам об Agile-методологии разработки и о том, что означает это слово – Agile (гибкий). Возникло ощущение, что каждый под этим термином понимает несколько разные вещи. Расскажите, как вы пришли к Agile-методологии, какие проектные задачи она помогла решить и какие проблемы вызвало ее применение – ведь ни одна методика не может быть панацеей. Нам бы очень хотелось узнать, что вы думаете по этому поводу и что именно означает для вас понятие Agile.**

Скотт: Хорошо. На сайте, посвященном Agile-моделированию (*http://www.agilemodeling.com/*), есть мое определение этого понятия. Оно как раз у меня открыто, давайте прочту: «Организованная Agile-разработка программного обеспечения представляет собой итеративный и инкрементный (эволюционный) подход к разработке ПО, реализуемый в тесном сотрудничестве самоорганизующимися командами при эффектив-

ном управлении с минимальными формальностями, способствующий разработке высококачественного ПО в нужные сроки с наименьшими затратами, с учетом изменяющихся потребностей заинтересованных сторон».

Здесь, конечно, много громких фраз. Но, по-моему, это очень важные параметры, и очень во многом Agile-разработка основывается именно на командной работе. Так например, ко всему, что касается тесного сотрудничества или самоорганизующихся команд, нужно подходить с точки зрения командной работы. Но один из аспектов, о которых многие из тех, кто использует Agile-методы, к сожалению, не говорят, – это управление.

Ни одна команда не работает в вакууме. Всегда есть рабочая среда. И работой этой команды необходимо управлять, и такое управление в некоторых аспектах может быть полезным. К сожалению, оно не всегда осуществляется правильно, поэтому некоторые программисты воспринимают в штыки само понятие *управления*. Каждый хочет, чтобы его было как можно меньше и чтобы оно при этом было как можно более эффективным.

Очень важным параметром является работа в тесном контакте с заинтересованной стороной, а это тоже аспект командной работы. В рамках Agile-методики мы говорим о концепции «полной команды». Можно считать или не считать это методикой, как хотите. Чтобы команда могла выполнить свою задачу, она должна обладать необходимой квалификацией, кроме того, для этого ей необходима совместная работа. И это очень хорошо. Но часто упускают из вида, что кто-то должен следить за работой всей команды, убеждаться, что команда работает в правильном направлении и то, что она делает, нужно всей остальной организации. Мне кажется, этот вопрос не всегда поднимается в среде Agile-разработки.

Эндрю: **Эти два понятия – управление и самоорганизация – кажутся мне прямо противоположными.**

Скотт: На самом деле, это не так. Каждым человеком что-то руководит, об этом часто забывают. «Самоорганизация» не означает, что вы абсолютно бесконтрольны и делаете что хотите. Это означает, что участники команды сами решают, как они будут достигать своих целей. Но эти цели, используемые ресурсы, сроки выполнения – все это определяется организацией. Кто-то оплачивает счета. Уже финансовый вопрос сам по себе представляет сложность, ведь если организации не понравится работа вашей команды, она может перекрыть вам кислород. Это жесткий, но вполне реалистичный механизм регулирования. С чего начинает работать команда? Если это проект разработки ПО, должны быть заданы

некоторые цели, которых вы должны будете достигнуть. Необходима исходная концепция, видение. И определение этой концепции, миссии – также одна из форм управления. Права и обязанности в группе, структура подчиненности – все это относится к управлению. Например, у младшего программиста может быть меньше права голоса в решениях, чем у старшего технического руководителя. И это тоже один из аспектов управления.

То есть очень часто управление не берут в расчет. Это факт, просто мы редко об этом задумываемся. У нас, команд, применяющих Agile-методы разработки, настолько сильна паранойя по поводу бюрократии и бесполезных формальностей, что все связанное с управлением немедленно принимается в штыки: «Нам не нужно управление, это бессмысленная трата времени!» Вовсе нет. Плохое управление может быть помехой. Так давайте стремиться к эффективному управлению, поскольку его задача в том, чтобы помочь нам принимать верные решения и двигаться в нужном направлении. Давайте не будем выплескивать вместе с водой и ребенка.

Эндрю: **То есть командой могут управлять общая концепция, цель проекта, а также знания и ограничения, определяемые самой командой и компанией. Но я видел немало проектов разработки, которые вышли из-под контроля, потому что это проект управлял концепцией, а не наоборот. Происходило так называемое расползание границ проекта.**

Скотт: Да. Концепция должна со временем эволюционировать, и это правильно, поскольку меняется и сама ситуация. На деле бывает так: задолго до фактического запуска проекта у кого-то появляется идея, что вы должны сделать то-то и достичь каких-то целей на основании таких-то деловых причин. У любой организации всегда есть гигантский список потенциальных целей. Но финансирование всегда ограниченно, поэтому приходится определиться, чего именно они хотят, и делать это по возможности эффективно. Это обоснование проекта направляет и регулирует работу, поскольку необходима уверенность, что вы двигаетесь к нужной цели. А если цель меняется, вам нужно удостовериться, что изменения должным образом отражаются на проекте, и вы, конечно, хотите сами управлять такой эволюцией. Такую сверку с реальным положением вещей нужно проводить как можно чаще, на деле же она выполняется гораздо реже, чем следует.

Пару лет назад я проводил исследование в журнале «Dr. Dobb's»[1] на тему того, как люди определяют успех. По-моему, если у проекта про-

[1] «Dr. Dobb's» – ежемесячный компьютерный журнал. Издается в США компанией CMP Technology. – *Прим. перев.*

блемы, кто-то должен прямо сказать: «Ребята, у проекта проблемы!» Кто-то должен принять решение о том, как вытаскивать команду. Если никто не знает, как справиться с проблемой, нужно действовать решительно, прекратив пускать деньги на ветер. Но это мое мнение, и я не знал, сколько еще людей его разделяет. Поэтому мы провели опрос. И только 41 или 42% респондентов заявили, что в их организациях, если у проекта начинаются проблемы, отмена такого проекта считается успешным шагом, что печально. Если я понимаю, что с проектом не все в порядке, то пытаюсь решить его проблемы как можно скорее. Либо останавливать его реализацию на максимально ранней стадии. Это не лучшая идея, но уж лучше потратить полмиллиона долларов и получить суровый урок, чем получить тот же урок, потратив три-четыре миллиона долларов.

Единственный способ сделать это действенным – следить за тем, что происходит, и задавать вопросы такого плана: команда все еще показывает успешные результаты? Есть ли еще смысл продолжать этот проект? Потому что сами команды могут не осознавать этого. И команда может показывать фантастические результаты в рамках заданных объемов. Но обстановка уже изменилась, а команда могла этого не знать. Какими бы ни были причины для запуска проекта, они могут измениться, и тогда проект необходимо остановить.

Например, осенью 2008 года у нас разразился финансовый кризис. Для многих финансовых компаний деловые условия изменились радикально в краткий срок с сентября по октябрь. Я уверен, что многие из них отложили некоторые проекты на дальнюю полку, потому что окружающая обстановка настолько изменилась, что эти проекты не имели больше смысла. Да, здесь нет никакой вины команды разработчиков. Но зачем же продолжать тратить деньги?

Эндрю: **Сама идея о привязке работы команды к общей деятельности компании, о том, что кто-то должен следить за тем, чтобы проект развивался в нужном направлении, напоминает мне о проблеме, с которой, полагаю, сталкиваются многие команды при Agile-разработке. Но я не слышал, чтобы кто-то ее обсуждал. Может, вы немного проясните ситуацию. Одна из обычных практик в Agile-командах – привлечь в команду представителя заинтересованной стороны. Иногда даже на полную ставку. Понятно, что внедрение в команду представителя заинтересованной стороны или деловых кругов может быть очень полезным для обмена информацией и удержания проекта в нужных рамках. Но, если честно, никогда не видел, чтобы это по-настоящему хорошо работало, поскольку я крайне редко встречал представителей заказчика, готовых всерьез тратить время на такую работу.**

А если команде действительно удается заполучить представителя заинтересованной стороны, боюсь, возникает та же проблема, поскольку и он перестает замечать, что происходит. Если включенный в команду представитель деловых кругов вместе с командой не видит нужных вещей, кто будет следить за целями проекта и проверять их на соответствие потребностям компании?

Скотт: Здесь есть две аспекта. Иногда пришельцы со стороны становятся своими. И тут неважно, кого приглашают в команду. Если вы достаточно тесно работаете с этой командой, то устанавливаете с ней связь, а связь с командой, откуда вы пришли (то есть, в нашем случае, с деловыми кругами), ослабевает. И это может стать проблемой.

Один из рисков концепции «единой команды» – групповое мышление. Основная идея этой концепции следующая: если группа работает вместе достаточно долго – например несколько месяцев – ее участники начинают мыслить сходно, переставая что-то замечать. Психологи иногда говорят об этом в контексте принятия рискованных решений. Например, вам не очень хочется ставить 100 долларов на скачках. Но один из ваших друзей ставит на лошадь 100 долларов, потом те же 100 долларов ставит еще один, и вот эта ставка уже не кажется столь рискованной. То есть группа вместе принимает решение, которое каждому кажется рискованным, но поскольку вы решаете всей группой, это ощущение риска сглаживается. Такое бывает.

То есть в командной работе есть очень интересная динамика, с которой необходимо считаться. И отслеживать ее – одна из задач управления. Проблема здесь – и именно поэтому, я думаю, люди так негативно относятся к управлению в сфере информационных технологий – в том, что многие организации в подобной ситуации переходят к командно-административному режиму работы. Люди, осуществляющие управление, считают, что они руководят проектом. Они пытаются направить его, слишком активно вмешиваются в его работу и непомерно обременяют команду. Они начинают требовать постоянного предоставления отчетов о ходе выполнения работ, или посещения ежемесячных заседаний контрольного комитета, или назначают по три собрания в месяц для обсуждения этапов и отчета о работе. Это ложится тяжким грузом на плечи команды. Хорошее управление заключается в том, чтобы обеспечивать возможности и мотивировать. Оно не должно быть командно-административным и не должно быть в тягость. Конечно, издержки будут, но их не должно быть слишком много. Если же их слишком много, управление, скорее всего, уже идет во вред, и с этим нужно бороться.

Дженни: **И что вы для этого делаете? Каков первый шаг к тому, чтобы понять, что команде, возможно, навязывается чрезмерное управление?**

Скотт: Нужно максимально автоматизировать все процессы. Мы с Пером Кроллом написали об этом полтора года назад доклад «Методика Lean в управлении проектами разработки». Его основные положения: автоматизировать процессы насколько возможно, максимально упростить процедуру подачи точных отчетов о показателях, а для того, кто осуществляющет управление, – понять, что разные команды поразному проявляют себя в разных ситуациях. Управляющий проектом должен быть гибким и понимать, что команда из пяти человек работает иначе, чем команда из пятидесяти. Команда, разрабатывающая базу данных, работает иначе, чем команда, разрабатывающая веб-сайт. Ваша команда разработчиков веб-сайта будет работать иначе, чем моя команда. Все мы разные, и команды у нас разные, даже если мы разрабатываем похожие вещи по одной технологии. Соответственно, и руководить этими командами нужно по-разному.

Очень распространенная ошибка среди тех, кто управляет командами, – попытка применить одинаковые методики и структуру управления к разным командам; это не работает, потому что для разных команд нужны разные способы управления. При этом цели могут быть теми же, разными будут пути, которыми вы пойдете к этим целям. Мне нравится следующая формулировка: нужно стремиться к повторяемости результата, а не процесса. Но это может стать непростой задачей для тех, кто привык к более жестким и бюрократичным методам работы.

Дженни: **Очень интересно. Когда вы говорите о разработке множества автоматизированных решений в этом контексте, мне представляется, что это может стать столь же обременительным, как и создание обширной документации. Создание инфраструктуры для тестирования, внедрение сценариев автоматизации и всех методов для автоматизации вашей деятельности, необходимые для этого ресурсы – все это кажется немалым трудом.**

Скотт: Очень хорошее замечание. Если вы собственноручно куете все свои орудия труда, это очень нелегкая работа. И это, мне кажется, еще одна из задач, стоящих перед Agile-разработчиками. Работа многих Agile-команд основана на ПО с открытым исходным кодом, и это весьма оправданно. Но и здесь есть ограничения, и одно из этих ограничений – вопрос интеграции, в частности, управления и точной передачи параметров. Здесь требуется выход за рамки традиции Agile.

Например, есть сэмпл «Джаз», и на сайте *http://jazz.net* можно скачать его демокопию. Но в Rational Team Concert, элементом которого является «Джаз», все это автоматизировано. Все эти вещи, столь дорогие сердцу Agile-разработчиков, их отчеты о тенденциях дефектов (defect trend) и графиков зависимости оставшейся работы от времени (burndown charts), или что угодно, о чем они отчитываются, все это делается

вручную, а тут они будут создаваться автоматически в режиме реального времени. Здесь нет большого секрета. Потому что это уже реализовано.

Если вам нужно выполнять все это самостоятельно, это невероятный объем работы. То есть у вещей, массово используемых сейчас в Agile-разработке. есть некоторые ограничения. А значит, есть задачи, которые необходимо решать.

Ни одной команде разработчиков в банке или страховой компании не придет в голову разрабатывать собственный компилятор. Такие вещи либо покупают, либо загружают бесплатно. Сейчас появляются все более эффективные комплексы средств сбора информации, необходимой для эффективного управления. Но, конечно, никто не горит желанием единолично и самостоятельно внедрять такие вещи, это, действительно, очень большой труд.

Эндрю: **Мы поговорили о препятствиях в разработке ПО, и хотелось бы завершить разговор на оптимистичной ноте. Расскажите, пожалуйста, о какой-нибудь замечательной команде из тех, с которыми вы работали.**

Скотт: Одной из лучших команд, в которых мне довелось участвовать, была команда, не связанная с разработкой ПО. Это была моя группа по карате. Я занимался карате около 10 лет, пока меня не вывела из строя травма, что, конечно, печально, но так уж случилось. В этом спорте есть очень интересные принципы, которые могли бы пригодиться многим командам. Так, в качестве структуры или правил в карате в Северной Америке применяется понятие поясов: вы начинаете с белого пояса, затем получаете желтый и так далее, вплоть до черного. И одно из правил таково: любой человек, имеющий более «низкий» пояс, то есть менее опытный, всегда может обратиться за помощью к тому, у кого более «высокий» пояс. И тот человек обязан приложить все силы, чтобы помочь ему. Эта помощь может быть, помимо прочего, и такой: «Я не знаю, как это объяснить, но нам может помочь вон тот человек; давай обсудим это с ним и научимся делать это лучше».

Желание и готовность обратиться за помощью очень важны. Но и готовность оказать помощь не менее важна; один из принципов боевых искусств – обучение и помощь – позволяет научиться большему, чем при самостоятельной отработке.

Даже если на тренировке присутствуют только оранжевые пояса, вы можете отрабатывать какие-то элементы вместе, помогая друг другу достигать поставленных целей. И вот эту готовность работать друг с другом, которой меня научило карате, я стараюсь применять и к командам разработчиков. Любопытно, что в среде Agile-разработки применяются

свои ката[1] программирования и имеются додзе[2] программирования. Мне кажется, многие переносят некоторые идеи из сферы боевых искусств в сферу разработки ПО. Боевые искусства возникли очень давно, и за это время те, кто их практикует, пришли к пониманию того, как можно учить людей. Потому что там все в основном на добровольных началах: будучи взрослым, вы начинаете заниматься боевыми искусствами, чтобы научиться либо чтобы просто быть в форме, все зависит от ваших целей. Но в любом случае вы делаете это для самосовершенствования.

Эндрю: **Это очень интересная для меня тема. Я около десяти лет занимался другим японским боевым искусством — айкидо. И что мне действительно нравится в этом виде спорта — совместные тренировки. И то, что важной частью тренировок является обучение более юных бойцов, особенно когда сам становишься старше и опытнее. Совершенно неожиданно для себя я сделал открытие: обучая других, я сам учусь даже больше, чем когда кто-то учит меня.**

Думаю, это еще один вывод, подходящий для программирования и разработки. Например, когда мы с Дженни писали учебник по программированию на C# в серии «Head First», я очень многому научился, пытаясь объяснить некоторые идеи новичкам. То есть, приступая к этому проекту, я, конечно, понимал, например, основные принципы объектно-ориентированной разработки. Но попытки объяснить начинающим программистам на C#, для чего нужна инкапсуляция или чем интерфейсы отличаются от абстрактных классов, причем объяснить так, чтобы они действительно смогли понять и применить эти знания, вывели меня на совершенно новый уровень понимания.

И я понял, что все это напрямую связано с моей работой. Я обнаружил, что будучи руководителем, особенно когда в мою задачу входит научить людей в команде и помочь им не только делать свою работу, но и развиваться профессионально, сам многому у них учусь, зачастую безотчетно, но иногда и осознанно. Иногда тот, кто программирует всего несколько лет, генерирует отличные, совершенно новые идеи. Поэтому я очень хорошо понимаю, о чем вы сейчас говорите.

Скотт: Это интересно, особенно если посмотреть на исследования разных сочетаний в парном программировании. Например, выяснилось,

---

[1] Ката — последовательность движений, связанных друг с другом принципами ведения поединка с воображаемым противником или группой противников. — *Прим. перев.*

[2] Додзе — места, где проходят тренировки, соревнования и аттестации в японских боевых искусствах. — *Прим. перев.*

что если ставить в пару новичка и профессионала, то они оба выигрывают от такого сочетания. Понятно, что новичок многому может научиться, работая с профессионалом. Но и профессионал учится, отвечая на его вопросы. А вопросы иногда бывают очень точными: «Почему ты это делаешь?» Ну, потому что... постой-ка... действительно, *почему* я это делаю? Это заставляет задуматься о своих действиях, что дает возможность совершенствоваться. «Действительно, это не очень разумное действие, и я, пожалуй, могу делать это по-другому и лучше».

Эндрю: **Надо же. Вы сейчас сказали то же, о чем мы с Дженни писали в нашей первой книге. У меня даже есть с собой этот текст — это отрывок из главы о парном программировании и сочетании новичка и старшего специалиста: «Нередко молодой участник команды задает, казалось бы, наивный вопрос о программировании, который, как выясняется, выявляет серьезную проблему. Особенно часто это происходит с проблемами столь давними, что старший специалист уже не замечает их, принимая как данность. Иногда масштаб проблемы с кодом выявляется только когда ее начинают кому-то объяснять». Просто один в один то, о чем вы сейчас сказали.**

И это вызывает у меня один вопрос: почему так сложно заставить программистов применять эту методику? Из всех методик, гибких или еще каких-то, которые я применял в своих командах или о которых говорил с другими, именно парное программирование было нереально сложно внедрить. Заставить программистов работать в паре посложнее, чем обязать их выполнять автоматическое модульное тестирование или разработку на основе тестов.

Дженни: **Мне кажется, у людей есть ощущение, что, когда двое занимаются одной и той же работой, это лишняя трата ресурсов, и именно поэтому они избегают эту методику.**

Скотт: Часто так и есть. Кроме того, людям не очень уютно в паре по сравнению с тем, когда сидишь за своим столом один и работаешь самостоятельно. Что интересно, при парном программировании невероятно устаешь. Работаешь пять или шесть часов и выдыхаешься полностью — потому что все это время действительно работаешь.

Один из способов, хоть и несколько жесткий, — принудительное применение методики. Обычно я делаю так: подаю идею команде — вот в чем ее суть, вот как это делается. Но это непросто. Мы проговариваем все аспекты и обсуждаем сложности. И я добиваюсь того, чтобы команда согласилась попробовать поработать так в течение месяца. Мы меняем состав пар каждый день, приходится работать с кем-то новым, а не с тем, с кем работал вчера. И проговариваем все проблемы. Весь месяц одиночное программирование запрещено. А в конце месяца решим, хотим

ли мы продолжать работать в таком же режиме. И вот что я обнаружил: если принудительно внедрить практику, заставив команду поработать в таком режиме, то к концу месяца очень немногие хотят вернуться к одиночному программированию. Но требуются усилия и время. Нужен труд, чтобы вытащить эту «рыбку», наверняка есть и другие способы, но в данном случае нужно просто браться и пробовать. И с парным программированием – какое-то время необходимо вынуждать людей так работать. Потому что на первых порах работать так *действительно* неудобно. Чувствуешь себя не в своей тарелке. Нужно просто начать это делать. По моему опыту, в командах, которые попробовали эту методику, очень немногие участники – может быть 5% – хотят затем вернуться к одиночному программированию. Но для этого требуется месяц работы.

Эндрю: **Думаете, это из-за того, что человеку трудно представить, как что-то работает на практике, если он никогда прежде этого не делал?**

Скотт: Думаю, да. Парное программирование изначально сложная методика, но у нее много преимуществ. Есть внутренние преимущества, которые сложно оценить извне, в этом и сложность. Здесь очень просто сказать: «Двое за одним рабочим столом – значит, они вдвое менее продуктивны». И легко отбросить идею, которую никогда не пробовал. Но стоит только получить какой-то опыт в этом, как вы поймете, что она вполне работает.

Эндрю: **Вам не кажется, что это общее правило принятия изменений командой? Что ее участники, попробовав что-то, уже не захотят возвращаться к работе в прежнем режиме, хотя поначалу будет много стонов и жалоб?**

Скотт: Общий вывод, по-моему, таков: если есть что-то, о чем много говорят, и есть свидетельства того, что это работает, стоит попробовать. Не уверен, что вы уже подошли к такому возрасту, когда понимаешь, что полезно есть отруби, – но, поверьте, вы тоже к этому придете! Это не то, с чего всем хотелось бы начинать. Но в конце концов думаешь: «Хм, пожалуй, это полезно». То есть нужно просто перебороть себя и начать делать.

27

Скорость или качество

Почему нам приходится выбирать?

Джоанна Ротман

В одном из проектов, которым я руководила, у меня состоялся следующий диалог со старшим менеджером компании:

Большая шишка: Прекратите эти проверки кода. Они тормозят проект.

Джоанна: Но мы тогда не узнаем, где в нем баги. Нам нужны проверки кода.

Большая шишка: Прекратите их, или я вас уволю.

Джоанна: Вы уволите меня за то, что я делаю все, как надо?

Большая шишка: В данном случае надо закончить проект как можно скорее. Прекратите проверки кода.

Довольно странная беседа, да?

Мне бы очень хотелось сказать, что этот случай был особым. Но даже несмотря на то, что конкретно эта беседа состоялась более 20 лет тому назад, подобные ей происходят то здесь, то там ежедневно. Это история о противостоянии команды давлению руководства, заставляющего ускорить работу. Команда знала, что отличную работу *можно* сделать быстрее, только делая ее правильно.

Как мы к этому пришли?

Вернувшись на машине времени к началу проекта, мы увидим, как Большая шишка собрал команду и сказал: «У меня отличные новости.

Я продал новую версию приложения по управлению процессами. Однако вот эти три вещи оно должно делать быстрее, и в нем должны появиться пять вот таких новых особенностей. И нам это нужно сделать за полгода».

Проектная группа из шести разработчиков собралась вместе обсудить проблему. Они работали командой уже несколько лет. «Новичок» команды уже полтора года работал в ней над продуктом, а ее «старожил» приступил к нему четырьмя годами раньше. Они знали друг друга и знали, как работать вместе.

У приложения ProcessControlApp не было графического интерфейса пользователя, поскольку оно было интегрировано в производственную линию. Доступ к приложению осуществлялся посредством командной строки или API. ProcessControlApp визуально проверяло содержимое строки, принимало по нему решение и выводило строку по этому решению. Таким образом, даже не обладая графическим интерфейсом пользователя, ProcessControlApp было сложным приложением, в котором имели значение и производительность (как быстро может программа принять решение?), и надежность (сколько времени проработает программа?), и точность (сколько неверных положительных и отрицательных решений принимает программа?).

Над предыдущими выпусками ProcessControlApp они работали вместе, подключив двух тестировщиков и одного писателя. Они знали, как совместно работать в команде. Они сказали вице-президенту по программному инжинирингу, что им нужны те же два тестировщика и писатель.

Я была менеджером по гарантии качества программного обеспечения, то есть должна была направлять тестировщиков на различные проекты. Я тогда встретилась с вице-президентом по инжинирингу Нэнси и объяснила, что у меня нет свободных тестировщиков. «Но вам придется предоставить им двух тестировщиков. Иначе они не смогут закончить проект».

«У меня нет тестировщиков. Смотрите, тут видно, кто где занят». Мы обсудили другие проекты и их относительные приоритеты [1]. Нэнси согласилась, что я и вправду не могла предоставить тестировщиков. Я предложила нанять еще кого-то, но Нэнси ответила, что этого не позволяет бюджет.

Подумав минуту, я сказала: «Хм, я не могу тестировать сама и не могу дать им тестировщиков. Но кое-что для команды я сделать могу. Могу руководить их проектом, помогая им находить варианты и устранять препятствия, чтобы они могли нормально работать. Поскольку это приложение для управления производственным процессом без графиче-

ского интерфейса пользователя, это возможно. Сложно, но возможно. Вы не против?»

«Поговорите с командой, не против ли они».

О команде

Эта команда была настоящей командой, а не просто группой [2]. Команда применяла собственный подход, что, отчасти, и сделало ее столь успешной. Но в наибольшей степени залогом ее успеха было доверие между членами команды.

Хотя я и была руководителем проекта, команда доверяла мне не совсем так же, как друг другу. Они с радостью выслушивали мои предложения по их работе, но никогда не принимали их не глядя. Они тщательно обсуждали каждую идею. Если они ее не отклоняли, то адаптировали так, чтобы она работала на них.

Стать частью команды

Я созвала собрание команды на следующий день и объяснила: «Я не могу дать вам тестировщиков; я никого не могу вам дать. Я не могу и сама заниматься тестированием – мне все еще нужно руководить. Но я могу предложить вам свои услуги в качестве менеджера проекта, особенно если вам нужен кто-то, кто бы устранял помехи и другие возникающие препятствия».

Это было мое первое собрание с этой командой. Я знала этих ребят и раньше – некоторых больше 10 лет, неофициально. Вот каким я запомнила это собрание.

Дэн, один из старейших членов команды и один из основных разработчиков, сказал: «Что ж, Джоанна неплоха как тестировщик и еще лучше как менеджер проекта, но нам и вправду нужны тестировщики, а не еще один руководитель. Прости, Джоанна». Дэн одарил меня своей медленной улыбкой и снова погрузился в кресло.

Фред согласился: «Джей Эр, очень мило, что ты предлагаешь, но нам реально нужны тестировщики». Фред тоже был одним из основных членов команды. Его пальцы барабанили по столу, а колено подрагивало. Фред был талантливым разработчиком, и обладал рядом раздражающих привычек.

«Но я не могу дать вам тестировщиков. Нет ни одного свободного, и денег, чтобы нанять кого-то еще, тоже нет. Я предложила стать менеджером проекта не потому, что считаю, что вам нужен руководитель, а чтобы помочь вам придумать, как работать в непривычной для вас ситуации. Понимаю, что ваша команда выработала отличный процесс,

который раньше работал. Но тот процесс зависел от тестировщиков. А их я предоставить вам не могу».

Клайд спросил: «Почему? В чем проблема?»

Я вытащила листок с распределением тестировщиков, оставшийся после беседы с вице-президентом, и объяснила, как нужны были тестировщики в других проектах, и что этот проект, хоть и был важным, но менее приоритетным, чем другие.

Дэн нахмурился: «Это просто глупо».

Я выступила вперед. «Я уже проверяла с вице-президентом и Большой шишкой. Этот проект шестой в списке, а тестировщиков хватает только на первые пять. Я не пытаюсь испортить вам жизнь. Понимаю, что это звучит как „я из внутренней налоговой службы и я вам помогу“. Кроме того, что я из руководства.

Но ваши ребята меня знают. Вы в курсе, что я много знаю о проектах и не собираюсь учить вас разработке. Я могу помочь вам найти способ справиться с этим проектом. Но только если вы согласитесь с моим предложением».

Дэн откинулся в кресле и громко выдохнул. «Ладно. Почему бы тебе не выйти ненадолго, пока мы тут все обсудим?»

«OK, я буду в своем кабинете. Позвоните мне туда».

Минут через пятнадцать мне позвонили, и я вернулась в переговорную.

Фред сказал: «Джей Эр, команда все обсудила, и мы тебя берем. Только запасись отличными идеями. Тут никто не уверен, что ты нам нужна».

«Отлично. Если я вам буду не нужна, увольте меня. Ну, не на самом деле увольте, просто скажите, что отказываетесь от меня, и я перестану помогать. OK?» – спросила я.

Все кивнули или сказали «О'кей».

Мы поговорили о том, как они хотели работать и чем я могла помочь. В мои задачи входило: назначать время собраний, находить конференц-залы для проверок, а также придумывать, как организовать парочку циклов с тестировщиками, которые работали над предыдущими проектами, чтобы понять, могут ли они исправить регрессионные тестовые скрипты.

«Как могли испортиться регрессионные тестовые скрипты?» – мне хотелось знать, что произошло, тогда я могла бы понять, сколько времени нужно тестировщикам, понять, возможен ли такой вариант вообще. Один из самых молодых членов, Сэм, покраснел и объяснил, что он улучшил глобальный класс, и теперь кое-что не работало. Он просмотрел регрессионные тесты, но не понял, как работают хотя бы некоторые.

«Так, может, мне стоит узнать, не могут ли тестировщики потратить час на то, чтобы объяснить всей команде, как устроены регрессионные тесты? А потом уделять вам какое-то время, разбираясь с проблемами».

«Да, возможно, это сработает».

Верное начало

У этой команды было больше шансов, чем у любой другой. Они знали, как работать друг с другом в качестве разработчиков. Это были талантливые люди, каждый из которых хотел работать хорошо. И они были открыты любому способу работы, предъявляемому проектом, чтобы делать работу хорошо, отвечая запросам Большой шишки.

В команде царила культура «джентльменства». Они не ругались, когда все шло плохо. Они выпускали пар шутя. Если кто-то совершал глупую ошибку, можно было сказать ему: «У тебя, что, мозгов нет?» Но ругаться или опаздывать на собрания не разрешалось.

Технически, были у них и другие активы. Их автоматизированный набор регрессионных тестов работал на самом последнем выпуске. У них была автоматизированная система сборки. И они были знакомы с проверками кодов, модульным тестированием и постоянной интеграцией.

Перед тем как приступить к работе над новыми функциями, команда решила исправить набор регрессионных тестов, чтобы знать, работают ли текущие сборки, без ручного тестирования.

Разработчики и их прошлые тестировщики договорились о собрании по регрессионному тестированию.

Джек – тестировщик – начал объяснять, как работали регрессионные тесты. «Так, все смотрим на третью страницу, там находится центральный цикл тестов».

Сэм перебил: «Нет, это не центральный цикл. Взгляните на шестую страницу – вот центральный цикл».

Тут раздался голос второго тестировщика – Джил: «Нет, вы просто не понимаете, как это работает. Гляньте на основную часть второй страницы. Видите последовательность?»

Сэм посмотрел на Джил: «Думаешь, у меня мозгов нет? Любой дурак поймет, что цикл на шестой странице».

Джил посмотрела на страницу 6 и сказала: «А, теперь понятно, почему ты так думаешь. Посмотри на этот вызов здесь и вот на этот цикл там», – при этом она указала на разные страницы.

«О, теперь хоть что-то прояснилось. А почему об этом нет ни слова в ваших комментариях?» – спросил Сэм.

«Потому что у меня не хватало времени, и я думала, что буду работать с вами над следующим проектом. Еще не бывало, чтоб на такие проекты компания не давала тестировщиков», – ответила Джил.

Все обернулись ко мне. «Эй, мне жаль, но я не могу поставить на этот проект несуществующих людей. Джек и Джил, вы знаете, что ваш проект доходнее». Они кивнули. «Ладно, эта команда разберется, как выполнить эту работу, хоть ситуация и не из легких».

Тестировщики прошлись по работе регрессионных тестов, прояснив замысловатое место, столь озадачившее Сэма. Команда сформировала общий взгляд на работу регрессионных тестов и разделила их на несколько частей, распределив их между членами команды. Сэм решил заняться тем коварным местом, чтобы «набраться опыта с этой тестовой конструкцией», а тестировщики согласились отвечать на вопросы, если таковые появятся.

Через несколько дней у разработчиков работали все регрессионные тесты. Кроме того, они разделили их так, что могли по-разному оценивать исполнение и проверять, не нарушают ли новые функции что-нибудь еще. Теперь пора было приступать к новой работе.

Казалось, все идет гладко, – ровно до тех пор, пока не дошло до внедрения первой функции. И тут все регрессионные тесты дали сбой.

Командное решение проблем

Проще всего было обвинить разработчика – Фреда, который внедрил эту функцию. Но в команде это было не принято. На собрании все дружно заговорили о «безмозглом коде». Я предложила тут же на собрании быстро проанализировать причины, чтобы найти источник ошибки. Безусловно, несколько вопросов нуждались в проработке, и стоило потратить время на то, чтобы понять, что происходит. Команда согласилась.

Команда выяснила несколько моментов (рис. 27.1):

- Никто не видел кода до его проверки.
- Команда перестала применять свою многоуровневую систему регистрации изменений.
- «Слишком много» людей регистрировали изменения в один день.
- В набор регрессионных тестов никто ничего не добавил для проверки работы новых функций.

Дэн взял карандаш. «Ну, ребята, что будем делать?»

Фред ответил: «Слушайте, вы же знаете, что со мной бывает». Все, кроме меня, с усмешкой кивнули. «Я, правда, хочу, чтобы кто-то всегда

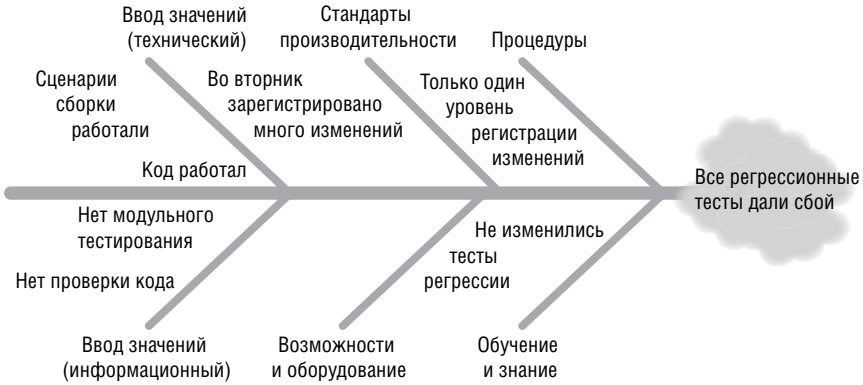


Рис. 27.1. *Диаграмма анализа причин*

проверял мой код. Я буду писать модульные тесты, раз уж у нас нет тестировщиков, – или есть, Джей Эр?»

Я ответила: «Нет, нету. Модульные тесты – это хорошая мысль. Не то же самое, что системные тесты, но помогут избежать этой проблемы».

Дэн сказал: «Но мы еще не закончили. Простых проверок кода и модульных тестов недостаточно. Нам придется пойти на многоуровневую регистрацию изменений, как в прошлом проекте».

Я ответила: «Вам это не нужно. Вы можете просто регистрировать их все время, а модульные тесты и тесты сборки пусть сами обнаруживают проблемы».

«У тебя, что, мозгов нет?» – спросил Сэм. «Это никогда не будет работать. Нам нужна многоуровневая регистрация».

«Хорошо, давайте я объясню, как это работает», – начала я.

«Ладно, может, вы и делали так в команде с тестировщиками, но без тестировщиков ничего не получится», – сказал Сэм.

«Ну, если ты так уверен, то конечно», – ответила я.

«Клянусь».

«А что остальные?» – остальные кивнули. «Хорошо, прежде чем перейти к альтернативам, давайте рассмотрим, как работает многоуровневая регистрация, хорошо? И составим график по методу набегающей волны [3], чтобы отслеживать все и знать, что происходит».

Сэм был обеспокоен: «Что такое график по методу набегающей волны и чем он мне поможет? Мне придется делать что-то еще, кроме своей работы?»

«Ну, единственное, что от всех вас потребуется, это потратить 30 минут на первоначальную подготовку графика. Потом мы будем обновлять его по информации из ваших отчетов о состоянии и моих личных встреч с вами. Если возникнут трудности, нам нужно будет еще полчаса провести вместе, вот и все. Устраивает?»

«А, ладно».

«Знаешь, Сэм, у Джей Эр есть мозги, по крайней мере, до сих пор были», – заговорил Фред, и все улыбнулись. – «Ты ведь еще не работал с ней, так?»

«Нет».

«Она довольно редко посещает нас, но она устраняет препятствия. Если Большая шишка тебя о чем-то попросит, предложи ему – только вежливо – пойти к Джей Эр, и она обо всем позаботится. Кроме того, если нам не понравится то, что она делает, мы можем ее уволить. Правда, Джей Эр?»

«Вы обещали!»

Теперь у команды были меры безопасности (многоуровневая регистрация изменений), чтобы поддерживать независимую одновременную регистрацию. Решили снова ввести визуальную проверку кода. И кроме проверки кода все разработчики теперь должны были писать модульные тесты для всего своего кода.

В набор регрессионных тестов системного уровня команда решила ничего не добавлять. Невозможно было выделить время и на разработку, и на это тестирование. Вполне хватало модульных текстов для всего своего кода.

Как выглядит проверка кода

Команда решила применять модифицированную проверку кода по Фагану. На каждой проверке кода был модератор и несколько проверяющих читателей, а сам автор выступал в роли секретаря. В настоящей проверке кода по Фагану проверяющие читатели просматривали бы код заранее поодиночке, а потом зачитывали бы вслух на проверке. Но эта команда решила не зачитывать код вслух во время проверки. Модератор спрашивал: «Есть ли у кого-нибудь замечания по странице 1?». И каждый должен был ответить да или нет и пояснить вопрос. Как и в проверке по Фагану, читатели должны были передать свои небольшие замечания автору в начале проверки.

Одной из причин не зачитывать код вслух была экономия времени. Если бы разработчики сделали выбор в пользу зачитывания вслух каж-

дой строки кода, команде пришлось бы запланировать гораздо больше проверок, потому что все они отнимали бы существенно больше времени. При такой проверке команда могла просмотреть 50 страниц кода за одну встречу.

Члены команды поняли, что им необходимо доверять друг другу и сообщать, если кто-то не закончил чтение кода. В таких случаях проверка прерывалась на том месте, где все закончили чтение, а на следующий день назначалась новая.

Если у автора были сомнения по поводу того, как ему устранить проблему в коде, он решал эту проблему вместе с модератором.

Модульное тестирование

Разработчики знали, что нужно продолжать тестирование. Но, как сказал мне один из них: «Я еще не настолько опасен, чтобы быть тестировщиком. Я умею проводить модульные тестирования, но в системных тестах я не так хорош».

Мы с Нэнси и командой постановили взять на себя техническую ответственность за отсутствие достаточного тестирования на уровне системы. Но команда решила, что ни коим образом не уложится в сроки без модульных тестов для всего написанного кода.

Разработчики писали свои коды, а потом модульные тесты. Один из разработчиков стал проверять, достаточно ли модульных тестов для каждого класса и модуля, с помощью инструментов McCabe (у нас была бесплатная UNIX-версия). Он сказал: «Я пишу отличные тесты для всего, что работает. Но, похоже, обычно я пропускаю то, что не работает. А с McCabe у меня меньше шансов это пропустить».

Регистрация изменений

Я попыталась убедить команду использовать постоянную интеграцию, но они не поддались – по крайней мере, мне. Но команда разработала свой подход к постоянной интеграции, разбив ее на этапы своим способом. Выбрали три уровня интеграции:

Сырой

Код, прошедший компиляцию и модульное тестирование

Готовый

Прошедший проверку код со всеми необходимыми исправлениями

Золотой

Код, прошедший сборку и все имеющиеся регрессионные тесты

Все разработчики отвечали за компиляцию и модульное тестирование собственного кода. Если разработчик не успевал, он должен был сообщить это мне, чтобы я утрясла вопрос графика с командой.

Сборки

Команда решила запускать автоматизированную сборку ежедневно в полночь. Каждую ночь весь код, помеченный как «золотой», собирался, и по нему проводилось модульное и регрессионное тестирование. Результаты направлялись всем членам команды по электронной почте.

Код, в котором кто-то регистрировал изменение, помечался как «сырой». Все разработчики встраивали сырой код локально, чтобы точно ничего не испортить. После того как код проходил проверку, его помечали как «готовый». Когда у разработчика весь код был «готов», он становился «золотым».

Графики

На следующий день мы собрались только для того, чтобы разработать график по методу набегающей волны. По верху стены я отметила стикерами месяцы, а под ними в соответствующих местах – основные известные целевые ориентиры, такие как внутренняя демоверсия, первоначальная демоверсия для потребителя и выпуск для потребителя. Нужные функции я указала вдоль боковой стороны презентационной схемы в том порядке, в каком разработчики собирались их внедрять.

Я спросила: «Что нам нужно, чтобы сделать первые две функции?»

Дэн ответил: «Нет, Джей Эр, мы работаем параллельно. Ты неправильно ставишь вопрос. Правильно – спросить так: кто что будет делать и когда?»

«Да, пока мы не планируем весь этот ужасный проект полностью. Все эти стикеры нужны затем, чтобы убедиться, что у нас есть план „Б“, на случай если не сработает этот план. Записывая задачи на стикеры, помните, что их нужно дробить как можно мельче».

Каждый из них записал свои задачи на стикерах. Большинство задач были рассчитаны на одну-две недели.

«А теперь, ребята, трудности у меня».

«Почему?»

«Потому что теперь я долго не буду знать, есть ли трудности у кого-то из вас. Надо сделать задачи помельче. Каждый из вас ведь делает что-то ежедневно, так?»

Они кивнули.

«Ладно, тогда я запишу все это только на первую неделю. Если вы и про вторую неделю знаете достаточно, добавьте и это».

Когда они сверили первые две недели по стикерам, я пояснила: «Разумеется, я не собираюсь каждый день вас спрашивать, что вы делаете. Но если запланированный результат не получается, сообщайте мне. Причину, если знаете, тоже сообщайте мне. Какие-то задачи вы будете выполнять вовремя, какие-то раньше времени, какие-то – позже. Моя работа – понять, если вы опаздываете, как это влияет на работу команды и ход проекта. Если проблема системная, мы обсудим ее на собрании группы. Если нет, то только вдвоем.

Я не стану дергать вас, чтобы выяснить состояние дел. Вы сами будете отправлять мне отчеты по состоянию в пятницу утром, а по вторникам у нас будут индивидуальные встречи. Таким образом, я несколько раз в неделю буду узнавать, надо ли мне как-то реагировать. ОК?»

Сэм нахмурился. Он выглядел озабоченным. «Джоанна, мне кажется, что все это потребует от нас много работы. По крайней мере, от меня. Мне действительно придется дробить свою задачу на кусочки до конца проекта?»

«Я не знаю, как иначе управлять этим проектом при жестком графике и технических рисках. Есть предложение получше?»

«Нет».

«Ну, попробуйте сделать это на две недели вперед, а там посмотрим, насколько такая работа вас отвлекает. Если это будет занимать слишком много времени или вам покажется, что это пустое занятие, мы обсудим это на еженедельном собрании команды. Но у меня это раньше работало. Вам нужно детально планировать только ближайшие две-три недели. Не больше».

Отчеты о состоянии дел

По мере продвижения проекта члены команды отправляли мне отчеты о состоянии дел каждую пятницу утром. Я объединяла их и к полудню рассылала общий отчет, так что все члены команды знали, кто и что делает. Проблемы – например, вопросы производительности – мы обсуждали на собраниях команды.

Для Нэнси и Большой шишки я готовила ежемесячные отчеты. Первые три месяца все шло гладко. Но на четвертый мы столкнулись с рядом проблем по улучшению производительности программы и из-за них отстали. Вместе с отчетом о состоянии дел я отправила объяснительную записку, и мы продолжили.

Теперь быстрее!

Встретившись со мной наедине, Нэнси заметила, что команда отстает. Она сказала: «Что ты собираешься с этим делать?»

«А?»

«Ну, они отстают на неделю или две. Что ты собираешься делать, чтобы наверстать время?»

«Ничего».

«Что ты хочешь этим сказать?»

«Нэнси, команда работает так быстро, как только может. Они только над этим проектом и работают. Они делают все, что могут. Кое-кто из них уже работает сверхурочно, а это нехорошо, потому что до конца проекта еще шесть недель».

«А почему бы не сократить модульное тестирование и проверки? Это высвободило бы немного времени, так ведь?»

Я остолбенела. «Нэнси, ты что – шутишь? Без модульного тестирования у нас вообще тестирования не останется. Без модульного тестирования мы просто не сможем узнать, хорошо мы работаем или плохо. Ты ведь хочешь выпустить этот продукт через шесть недель, так?»

Нэнси кивнула.

«Тогда позволь моей команде работать так, как им это нужно».

«Ну, а как насчет проверок кода? Может, забыть о них? Сэкономили бы время».

«Нэнси, ты ведь уже давно не пишешь код, да?»

Нэнси кивнула.

«В общем, теперь это сложнее, чем тогда. Тебе когда-нибудь удавалось написать код, который бы сразу заработал?»

«Да, пару раз».

«Итак, твой код сразу заработал, допустим, пять раз за 10 лет, когда ты была разработчиком. Ты работала с указателями? Мне казалось, ты писала на ассемблере».

«Да, на ассемблере. Там указатели не такие, как в C».

«Тогда почему ты думаешь, что если убрать проверку кодов, то это позволит нам быстрее закончить? В итоге это лишь замедлит проект, когда мы не будем знать, где ошиблись. И почему ты все равно меня об этом просишь? В чем дело? На тебя давят?»

«Да. Большая шишка постоянно говорит мне, что если мы через шесть недель не предоставим продукт, то моя работа не продлится и восьми недель».

«А разве он тебе это не про каждый проект говорит?»

Нэнси усмехнулась: «Да, про каждый».

«Тогда почему сейчас ты ему поверила?»

«Я действительно хочу убедиться, что вы делаете все возможное».

«Послушай, если Джек и Джил смогут оторваться от своего второго проекта и вернуться к нам для системного тестирования, это бы ускорило проект. Но разработчики не могут работать быстрее, чем сейчас.

Я вот что тебе скажу. На следующем собраании команды я спрошу разработчиков, что они думают по поводу возможностей ускорения работы. В любом случае результат я тебе сообщу, хорошо?»

«Ладно, попробуй хоть что-нибудь сделать».

В поисках ускорения

На следующем собрании команды я спросила разработчиков, что они думают по поводу ускорения темпов работы. Первым ответил Дэн: «Ну, если бы ты дала нам тех тестировщиков...»

«Извини, этого не будет. Я проверила все с Нэнси и не могу снять их с текущего проекта. Другие мысли у кого-нибудь есть?»

Фред прочистил горло: «Мы можем работать сверхурочно и один день из выходных».

«Вам правда этого хочется?» – спросила я.

«Нет».

«Тогда давайте пока что не рассматривать это как вариант. Переработки на этом этапе – плохая мысль. Конечно, если бы шла последняя неделя, то, возможно, пригодилось бы и такое последнее усилие. Но сейчас мы только сами себе измучаем. Еще предложения?»

Тим сказал: «Я почти закончил со своей последней функцией. Фред, хочешь, я помогу тебе с производительностью? Может, у нас получится вместе над этим работать».

Фред согласился.

«Кто-нибудь еще заканчивает свою работу? Может, парная работа поможет?» Больше никто не заканчивал. «Ладно, продолжайте работать над небольшими задачами. Дайте знать, если будут трудности. Я проверю состояние дел в пятницу».

Потеря недели времени

К пятнице стало понятно, что Фред борется за улучшение производительности. Но прогресса он пока не добился. Тим только что закончил свою работу над функцией и был готов помогать Фреду. Я отправила отчет о продвижении дел Нэнси и Большой шишке. Чтобы помочь Фреду, в поисках вариантов перераспределения я проверяла все задачи. Почти очевидно ему требовалась еще и помощь Дэна, чтобы поискать другие решения.

Большая шишка заглянул в мой кабинет и сказал: «Джоанна, есть разговор». Он шагнул внутрь и навис над моим плечом.

Я взглянула на него и сказала: «О'кей, дайте мне хотя бы сохранить то, что я делаю».

«Нет. Бросьте все и слушайте меня».

Я обернулась и спросила: «В чем проблема?» И предложила посетителю сесть. Он покачал головой.

«У меня нет проблем. А вот у вас, напротив, большие проблемы. Прекратите эти проверки кодов. Они тормозят нам проект», – Большая шишка грозил мне пальцем прямо в лицо.

«Но тогда мы не узнаем, где в нем баги. Нам нужны проверки кодов».

Большая шишка еще ближе подошел ко мне. Он кричал: «Прекратите их, или я вас уволю!» Его лицо покраснело.

Я встала. «Вы уволите меня за то, что я делаю все, как надо?» Большая шишка был невысок, но и я ростом не вышла. Я доходила ему до плеча. Но так все равно было лучше, чем сидя на уровне его живота. Большая шишка сделал шаг назад.

«В данном случае надо закончить проект как можно скорее. Прекратите проверки кода. На следующей неделе я надеюсь увидеть больший прогресс».

Большая шишка вышел из моего кабинета. Я глубоко вздохнула и зашагала к офису Нэнси, однако там уже Большая шишка орал на нее. Я решила подождать, чтобы поговорить с ней.

Что делать дальше?

Я не совсем понимала, что делать дальше. Команда не могла прекратить модульное тестирование и проверки кода – я полагала, что это нас обезопасит. Но мне решительно не хотелось терять свою работу. Терять еще больше времени на этом проекте тоже не хотелось. Я решила поговорить с Дэном прямо сейчас. Дэн был неторопливый малый. Он думал, прежде чем говорил, и, казалось, мало что могло его потревожить.

Помимо того что он был основным разработчиком, с ним можно было обсудить проблему. Он бы не запаниковал.

Я отправила письмо Дэну, попросив выделить мне полчаса, чтобы разобраться в этой проблеме. Я бы пришла к нему по первому зову. За ленивым обликом Дэна скрывался супермозг. Он был из тех, кто скорее плывет, чем шагает, кто выглядит полусонным, но вдруг поражает вас блестящей идеей. Если кто и мог сообразить, что нам делать, так это Дэн.

Примерно 10 минут спустя неторопливо вошел Дэн. «Ой, спасибо, что пришел. Хочешь, пойдем в твой кабинет?»

«Нет, думаю, вся нужная информация у тебя здесь».

«Ты прав. Давай посмотрим. Я не хочу терять еще неделю. И не хочу отменять проверки кода – или модульное тестирование, – но Большая шишка завелся. На следующей неделе я должна предъявить ему некоторый прогресс».

«Он грозился тебя уволить?»

«Да».

Дэн помолчал минутку и нахмурился. «Ох. Обычно он пугает увольнением Нэнси. А с менеджером он в последний раз так и поступил».

«Когда это было?»

«Две недели назад».

Кажется, у меня серьезная проблема.

«Дэн, я не собираюсь менять то, чем занимается команда. Проверки кода и модульное тестирование нужны. Но мне бы не хотелось терять работу – хотя бы еще несколько недель. Как думаешь, что тут можно сделать?»

«Ну, мы могли мы проверить код Фреда всей командой и найти проблему. Фред добавлял в код инструменты – значит, он знает, где что-то тормозит?»

«Он говорил, что делал это, но я не проверяла. Может, он что-то упустил?»

«Да, может быть пора проверить инструменты. Возможно, придется серьезно изучить, где и почему, по мнению Фреда, код так тормозит».

«Дэн, давай поговорим, что происходит у тебя. Думаю, Фреду нужна подмога, а ты гораздо дольше него работаешь с этой базой. Если ты будешь помогать ему, что нам делать с твоей работой?»

Мы обсудили с Дэном варианты его работы и решили пока ее не менять. Дэн собирался спросить Фреда о тестировании производительности и предложить тестовую проверку и проверку кода. Дэн мог предложить

ему проверить инструменты. Но это было все, что Дэн мог сделать до нашей встречи с Фредом во вторник, – если только Фред сам не попросил бы о помощи.

Я все еще не могла успокоиться. Не каждый день старший менеджер грозит тебе увольнением. Но мне было ясно, что команда меня приняла, потому что я помогала их работе. Я не собиралась мешать им, но мне нужен был план.

Сохраняя верность

Я попала в ловушку, о которой часто слышала от менеджеров и руководителей проектов. Если продолжать делать то, что считаешь верным, то потеряешь работу. А если подчиниться желанию руководства – неважно, насколько неверному, – потеряешь доверие команды. При этом график все равно под угрозой.

Я выбрала верность себе и команде. По правде говоря, принять такое решение было просто. Если моя команда работает хорошо, то зачем мне мешать им?

Ключевой момент

Мы решили на следующем собрании команды проверить инструменты и поддержку производительности в коде. Я исполняла роль секретаря, поскольку не была знакома с кодом настолько, чтобы эффективно комментировать. При подготовке собрания я предложила команде читать код вслух, потратив на проверку часа два. Внизу письма я добавила примечание: «Помните, нельзя никому рассказывать, что мы все еще занимаемся проверками кода. Большая шишка оторвет мне голову». Тогда я думала, что шучу.

Мы продвигались с этой проверкой кода практически ползком. За два часа мы прошли 15 страниц из 30. У Фреда было несколько мыслей, как переделать код, но ни одной, как ускорить работу. За два часа наши головы подустали.

«Давайте сегодня остановимся на этом», – предложила я.

«Нет, мы можем закончить», – сказал Сэм.

«Нет, Сэм», – подал голос Тим, он же «Тихоня».

«Нет?»

«Ни в коем случае. Мы устали. Мы сконцентрировались на этом коде и инструментах, и это нас куда-то привело. Но просто тратить еще время прямо сейчас не поможет. Мы должны оставить это и сделать перерыв. Мне нужно прокатиться на велосипеде».

«Слушай, Тим, я знаю, что все твои большие идеи рождаются, когда ты на велосипеде, но...»

«Нет, я учусь. Как и ты. Джей Эр, можешь записать на нас переговорную завтра с 10 до 12? Думаю, мы сможем закончить. А если нет, решим, что делать дальше. Сэм, послушай меня. Ступай, поиграй с собакой, отдохни, прими душ или займись еще чем-то. Но не смотри ты больше на этот код. Понял?»

«Уф, конечно». Сэм выглядел пришибленным.

Когда Тим вышел, Дэн улыбнулся. «Сэм, ты еще не понял, что настоящий лидер команды – Тим? Пора бы».

Я записалась на переговорную и сообщила группе по электронной почте свои идеи, а также место переговорной для завтрашней встречи. Беспокоило то, что она располагалась рядом с офисами высшего руководства.

Вечером дома я рассказала мужу, как прошел день. «Милый, помнишь, пару недель назад я говорила тебе о беседе насчет проверок кода?»

«Угу».

«Так вот, завтра утром у нас двухчасовая проверка прямо рядом с офисом Большой шишки. Возможно, вернусь рано и начну искать работу».

«Что ж, тебя ведь не в первый раз увольняют, правда?»

«Я предпочитаю термин *освобождать от должности*. Так и есть».

«Да ладно. Просто начни сегодня же писать резюме. И позвони мне завтра в перерыв, если все еще будешь трудоустроена».

На следующий день мы собрались в переговорной и начали с 16-й страницы. Мы уже немного продвинулись, когда в 10:40 вломился Большая шишка.

«Джоанна, чем вы занимаетесь? Это похоже на проверку кода!»

«Правда? Мы пытаемся выяснить, как наверстать время по проекту. Плана пока нет, но скоро будет».

«Я рад, что вы образумились. Вы же понимаете, что я прав».

Когда он вышел, все расхохотались. Первым заговорил Тим: «Джей Эр, ты не играешь в покер, но умеешь врать, не краснея?»

«Я сказала правду – по крайней мере, отчасти. Мы работаем над планом. Просто применяем инструмент, который ему не нравится». Все снова засмеялись. «Ладно, вернемся к делу».

К полудню у нас было еще два предложения по инструментам и предложение по изменению кода. Фред и Тим продвигались шаг за шагом, координируя, кто и что делает, и обещали, что в конце дня сообщат мне о своем плане.

Потратив какое-то время на проверку графика, я пришла к мысли о поэтапном выпуске. Мы могли выпустить приложение с новыми функциями вовремя, а с улучшенной производительностью – двумя неделями позже. Я предложила команде рассмотреть эту возможность на следующем собрании.

Вот и успех

Команда согласилась на поэтапный выпуск. Они продолжали применять проверки кода, модульные тесты и многоуровневую регистрацию. Я следила за продвижением проекта. Закончив со своими функциями, остальные члены команды влились в работу Фреда с Тима над производительностью. Нам повезло. К назначенной дате у нас были готовы все изменения и по функциям, и по производительности.

На следующий день мы отметили это мороженым, попутно обсудив, чему же мы научились.

Сэм начал: «Джоанна, я и понятия не имел, что дробление задач действительно так сильно поможет. Но еще больше меня впечатлили проверки кода и модульное тестирование. Без них я бы никогда не справился».

Фред добавил: «Надо было мне предложить проверки раньше. Тогда бы обошлось без переработок».

Тим сказал: «Может, теперь ты будешь мне верить. Единственное, что мне осталось, – это заставить тебя перейти только на натуральную пищу». Все засмеялись.

«Меня по-настоящему поразило то, что всем вам – вместе или в одиночку – удается сделать, – добавила я. – Спасибо за возможность работать с вами».

Эпилог

Я провела в этой организации еще несколько месяцев, потом меня уволили (что было лучшей переменой в моей карьере за всю жизнь). Команда продержалась еще два года. Большая шишка разделил их, когда стало понятно, что они отлично работают вместе и влияют на другие команды.

С некоторыми из них я поддерживаю отношения. Кто-то все еще работает вместе. Двое переместились на руководящие должности, занимая которые они стараются внедрять опыт той великой команды. Я с нежностью вспоминаю работу с этой прекрасной командой.

А как же продукт? Он получил японскую премию за качество. Большая шишка принял награду, сказав: «Поражало стремление команды к качеству».

Ссылки

1. Rothman, Johanna, 2009, «Manage Your Project Portfolio: Increase Your Capacity and Finish More Projects», Raleigh, NC: Pragmatic Bookshelf.
2. Rothman, Johanna, and Esther Derby, 2005, «Behind Closed Doors: Secrets of Great Management», Raleigh, NC: Pragmatic Bookshelf.
3. Rothman, Johanna, 2007, «Manage It! Your Guide to Modern, Pragmatic Project Management», Raleigh, NC: Pragmatic Bookshelf.

28

Тяжко, да?

Марк Денович и Эрик Ренки[1]

Всего лишь пешка... в игре жизни, или «Почему такой цивилизованный человек, как ты, гниет в этой дыре?»

В августе 1998 года я получил по электронной почте ответ на свое резюме, размещенное мной на сайте поиска работы. Я не очень-то на него рассчитывал, видимо, не умея распознать, когда в жизни настает поворотный момент. Я сразу же позвонил, и вскоре мне перезвонил основатель компании, он же ее президент и исполнительный директор. Я и представить не мог, что всего через несколько дней взойду на борт вечно терпящей бедствие компании, став членом одной из самых крутых команд, какие только знал. Все мое прежнее образование меркнет перед с тем, что мне предстояло узнать за ближайшие шесть лет.

Слова имеют неприятную привычку возвращаться, настигая меня, и эта история явно не исключение. Было бы легче сказать, что все последующее – плод моего воображения, но я не очень-то хороший выдумщик. Я мог бы оправдаться, сославшись на свою действительно слабую память. Но с этим рассказом мне существенно помог тогдашний и нынешний мой коллега по работе Стэн Гранайт. Его юмор, уникаль-

[1] Хотя эта глава написана Марком Деновичем и Эриком Ренки (с помощью Стэна Гранайта, который пожелал остаться неизвестным), события изложены с точки зрения Марка.

ная точка зрения и потрясающая память (подпитываемая из архива электронной почты) сделали эту историю крайне точной. Он же – автор большинства прозвищ, вошедших в обиход компании. Чтобы хоть както подстраховаться, я использовал эти прозвища, придумывая по ходу дела другие, при необходимости. К сожалению, виновные все же могут себя узнать. Я знаю, что слова часто ранят, так что могу лишь надеяться, что когда они прочитают это и мысленно вернутся к тому времени, то, как и я, посмеются и простят меня.

За шесть лет, что я пробыл в компании, там проработало более 160 человек. Когда я пришел, в штате было шестеро или семеро при максимальном штате – 42. Текучка была поразительно высокой, кроме команды разработчиков, которую наш директор признавал наименьшим неизбежным злом. Но когда заводились деньги, их щедро тратили на новых вице-президентов и директоров, часто нанятых непонятным образом и с еще более непонятными стратегическими целями. Если приходилось туго, им же первым и указывали на дверь, обычно без особых церемоний. Но как бы то ни было, ядро отдела разработчиков оставалось на удивление стабильным. Начав с нуля, восемь разработчиков создали продукт для Windows, включающий 800 окон, 700 000 трок кода и базу данных из 750 таблиц и 800 хранимых процедур на 400 000 строк T-SOL. Все это было разработано без отрыва от работы над обычными проектами, во время которой тоже нельзя было расслабиться. В то время я был слишком наивен, чтобы реально оценить это достижение. Все мы были слишком наивны, возможно, это и был ключ к нашему успеху. Будь у нас хоть толика здравого смысла, мы сразу бы поняли, что все наши усилия никак не сочетались с курьезными решениями руководства и вытекающим из них преступно безобразным поведением наших заказчиков. В свою защиту скажу: я предполагал, что смело рисковать своими и инвесторскими деньгами можно только имея на руках выигрышную стратегию. Предположение верное, но я не догадывался, что тот же человек может делать самые бредовые вещи. Просто тогда мне еще не встречались люди вроде Хедли Ламарра.

Я увидел Хедли в офисном здании в деловом квартале Питтсбурга. Очень хорошо помню нашу первую встречу, в основном потому, что именно так себе ее и представлял. Он был хорошо одет, в костюме и галстуке, изысканно выражался и занимал респектабельный офис с хорошим видом из окна. Хедли рассказал, что Rock Ridge – компания, которая делает программы на заказ, главным образом для энергетики. Казалось, как бизнесмен он был определенно доволен собой. Где-то через час я с улыбкой вышел, уверенный, что снова удачно прошел собеседование. Телефонный звонок подтвердил это, и после недолгих переговоров о зарплате (выбил себе 20% сверх первого предложения)

я был принят. «Наконец-то настоящая работа». До того я немного подрабатывал, учась в школе, потом поступил в какой-то нелепый дотком, управлемый явным безумцем, а непосредственно перед Rock Ridge был низкооплачиваемым независимым консультантом. Меня не порадовали дресс-код (костюм и галстук), время начала работы (8:30 утра) и ежедневные поездки на работу, но я был счастлив, оставив позади все глупости, начать серьезную карьеру.

Но мой энтузиазм на новой работе быстро испарился. Мои немногочисленные коллеги были странными, но скучными. Я решил не замечать их. Дверь в многолюдный коридор постоянно и без предупреждения открывалась почти непосредственно в мой отсек. Но еще хуже был дьявольски отвратительный кофе. К тому же мне нечасто удавалось видеться с Хедли. Какие-то инструкции я получал в оживленных, но кратких беседах с занудливым (и заросшим) менеджером разработчиков, далее Админ. Как я понимал, моя задача состояла в создании пользовательского интерфейса и каркаса приложения для нового проекта – служебной системы учета на основе VB и Microsoft SQL Server. У меня не было опыта, чтобы заподозрить отсутствие у проекта нормальных требований, спецификаций и процессов. Так что, пока я не выяснил масштабы, в которых могу отлынивать, мой проект неплохо продвигался. Через две недели я познакомился со Стэном Гранайтом – тоже новичком, занимавшим соседний отсек. Пожимая ему руку, я и не догадывался, с кем имею дело. Пропорции этого парня никак не вписывались в офис. Он был гигантом. Ему явно было тесно в рубашке и галстуке, а особенно в крошечном отсеке, напоминавшем телячий загон. Когда его видавший виды компьютер зависал, я ожидал воплей в духе Халка или чего-то подобного. Я начал продумывать пути к отступлению. Но больше всего меня интересовало, что, черт возьми, этот парень тут делает. Стэн, похоже, думал о том же. Я расспросил его о первых впечатлениях:

> Хотя наша команда все-таки сплотилась, это произошло далеко не сразу. Есть старая пословица о браке: то что до свадьбы вы находили очаровательным, после приводит вас в бешенство. У меня по отношению к команде все было как раз наоборот: то, что позже немало развлекало нас, поначалу заставило меня усомниться, что я продержусь хотя бы три месяца.
>
> Когда я только начал работать в Оксфорд-центре, в комнате было шесть или восемь человек, упакованных в отсеки; больше всего это походило на формочку для льда, да и по размеру было немногим больше. После трехминутного инструктажа мне выделили компьютер, который чаще не работал, чем работал. Всеми этими фактами, впрочем, пренебрег Админ, который устроил мне к концу первой недели разнос, закончив его словами: «Я даже не понимаю, за что тебе плачу».

Я почти не общался с Хедли, пока мы не начали работу над BTK-версией Candygram, так что первое мое воспоминание о нем – это образец счета-фактуры в десятикратном размере, в котором было больше деталей, чем на картинах Иеронима Босха. Почти тысяча квадратных дюймов, касающихся исключительно формы, притом без всяческих данных. Кто, скажите на милость, набрасывает счет на грифельной доске и при этом считает необходимым нарисовать четко очерченный, хорошо сканируемый штрих-код PostNet и номера счета шрифтом MICR? Думаю, если бы он написал «тут штрих-код», мы бы и так поняли. Будь я повнимательнее, по всем этим утонченным формальностям без повода составил бы представление о его личности гораздо раньше, чем это в итоге произошло.

Теперь о моих коллегах. Монах Тук был полной противоположностью Админу, как по первому впечатлению, так и по последующему поведению. Поначалу впечатление было приятным: он по крайней мере признавал мое присутствие, чего нельзя сказать о Безумном Ляпнике (БЛ), который был вроде Харпо Маркса или Человека дождя (он ритуальным образом раскладывал на столе все свои ручки, маркеры и карандаши по типу и цвету, проверяя перед уходом, что все они лежат в идеальном порядке). Но после того как я полчаса пытался помочь Брату Туку вспомнить название замечательного нового ресторана, на который он недавно набрел (в итоге это оказалась одна из закусочных Chick-fil-A), у меня возникли подозрения. Вскоре я понял, что он обладает всеми выдающимися способностями программиста и отточенным логическим мышлением, чего нельзя было ожидать от нетехнаря.

Марк был склонен к крайне противоречивым утверждениям. Например, что убийство детей – не такое серьезное преступление, как убийство взрослых. Он объяснял: «Чтобы появился новый двухлетний ребенок, достаточно двух лет. Надо мной работали 25 лет, и я в начале своего экономического расцвета». И если Админ не проедал мне мозг, то беспрестанно повторял, что никто не сравнится с ним талантом. Я еще не научился извлекать удовольствие из их шутовства.

Думаю, отношение Админа поменялось однажды вечером, когда мы засиделись допоздна. Он продолжал потчевать меня сказками о возможностях своего мозга, объясняя, что у него четыре отдела мозга, и каждый способен самостоятельно работать над решением мировых проблем. Один из отделов, вероятно, осознал, что голоден, и он собрался пойти куда-нибудь поесть. Указав себе на голову, он добавил: «Этой штуке нужно много топлива для работы», а я ответил: «Админ, ты про мозг или про язык?» И впервые Админ утратил дар речи. Правда, всего на несколько секунд, но мне показалось, что после этого замечания он стал относиться ко мне не как к подчиненному, а как к равному.

Каким-то образом нам всем удалось пережить следующие полтора месяца, пока не открылся Центр разработки программ на юге Питтсбурга. Для нас здесь было с полсотни баров и ресторанов в десяти минутах ходьбы от офиса. А Хедли радовался, что технический персонал (поддержка) – не дотягивающий до стандартов Хедли даже при помощи дресс-кода – больше не будет позорить профессиональный образ, который он пытался создать в основном офисе. Но успокаиваться он не собирался. Стэн пишет:

> Большинство разработчиков были мужчины чуть за двадцать, и Хедли, решив, что нам не помешает надзор, назначил следить за порядком женщину на пару десятков лет старше нас. За свое кратковременное пребывание в Rock Ridge она успела ввести только «доску местонахождения», на которой, по ее словам, каждый должен фиксировать свое местонахождение и приблизительное время возвращения, если отлучается из в офиса. Когда кто-то просто хотел выйти за пончиком, это нужно было записать на доске. Доска местонахождения способствовала командному единству, поскольку ее дружно ненавидели все члены команды. То, что Марк полностью игнорировал доску, было для меня первым знаком, что с ним все в порядке. А также то, что его обед обычно состоял из банки алкогольного коктейля.

Примечание Марка: для потенциальных работодателей, избирателей или членов суда присяжных я хотел бы пояснить, что добавляя эти воспоминания, я не предполагал, что у Стэна стопроцентная память.

Примечание Стэна: Марк прав... отчасти. Иногда коктейля было на пару унций меньше.

Теперь, когда офисов стало два, компания собиралась вместе только по особым поводам (Рождество, часы скидок, периодические массовые увольнения). Первый час скидок застал нас всех в городе, в кафе Dingbats; компания тогда насчитывала от 15 до 20 человек.

Примечание Эндрю и Дженни: авторы познакомили нас со Стэном Гранайтом. Эндрю взял интервью у Марка и Стэна, и они рассказали еще несколько интересных подробностей.

Эндрю: **У меня два вопроса. Первый: как вы ухитрились написать столько действующего кода такой небольшой командой? И второй: как вы вообще за это взялись? Судя по всему, условия работы были тяжелыми, даже после переезда на юг. Похоже, для вас это стало настоящим испытанием. Но вы выдали огромный объем кода, и, как я понял, он реально работал.**

Стэн: Я отвечу на эти вопросы вкратце и, возможно, они отпадут. Во-первых, мы сделали все, потому что работали как бобры. Во-вторых,

команда не распалась, потому что каждый день был похож на представление комиков-импровизаторов. То есть просто играешь свою роль, и от этого весело. Сказал что-то не то – провалился.

Эндрю: **Но работать как черти это ведь еще не все. Я был в командах, где все работали как черти, но в основном ничего не получалось. Честно говоря, в некоторых командах я понял, что чем больше работаешь, тем хуже получается, потому что по большей части там приходится исправлять то, что сделано в последнюю минуту впопыхах и, естественно, не работает. Вы же, ребята, выдали такой продукт, который до сих пор ежедневно использует крупная энергетическая компания, для нее он жизненно важен и при этом работает достаточно хорошо. Так что работать как черти явно недостаточно. Я бы хотел узнать, чего, в таком случае, оказалось «достаточно».**

Марк: Думаю, дело частично в том, что наши умения не пересекались и не конкурировали, так что наши проблемы с самолюбием никак не влияли на код. К тому же самолюбие Админа было настолько велико, что его просто нельзя было сокрушить, так что здесь у нас была некая уверенность.

Не думаю, что у нас было много конфликтов и когда мы решали, что делать. У нас удачно подобрались именно те, кто хотел работать, и те, кто мог заглянуть чуть вперед и сказать: «Это фигня, надо сделать лучше. Если нет, будет плохо». Так что у нас была своего рода болевая чувствительность.

Стэн: И то, что ты говоришь про самолюбие, ни в коей мере не касалось кода. Даже Админ, объявляя себя величайшим SQL-программистом в мире и на Восточном побережье, мог держать удар. Например, он много их получил за ту историю со скрытым текстовым полем.

Эндрю: **Что за история?**

Стэн: Когда мы стали разрабатывать Candygram, уже была предыдущая версия 2.5. (Мы поговорим о ней позже, когда расскажем про Turducken.) Candygram 2.5 – приложение на Visual Basic 4.0. Мы с Марком посмотрели код и увидели скрытые текстовые поля, в которых хранилось общее состояние. Мы моментально подняли Админа на смех: «Разве ты не слыхал о переменной уровня формы?» – чем, конечно, сильно его достали. Явно нужно было почистить код. Он всегда оправдывался тем, что в Access 2.0 получалось только так: там был какой-то баг, и при использовании переменной в определенных условиях ее состояние терялось, что распространялось и далее.

Марк: Думаю, он просто ленился и поэтому использовал скрытое текстовое окно со встроенным связыванием данных, а иначе ему пришлось бы написать пару строку кода.

Стэн: Мы все и всегда были открыты для нового. У нас иногда были совместные сессии по проекту, где кто-то обязательно злился, но на него не реагировали. И даже не признавая этого, люди многому учились на совместных действиях. Чтобы это наладить, пришлось избавиться от Монаха Тука. Он тянул команду назад.

Марк: Да, думаю, это был важный момент. На пик мы вышли без единой паршивой овцы. С другой стороны, Монах Тук был дурак. Он был из тех, кто своей плохой работой тормозит весь проект. Если такой что-то сделает, то получается только больше работы: он не только мешает тебе выполнять свою работу, но и вынуждает наводить порядок в *его* работе. Думаешь, что он трудится над чем-то стоящим, пока не приходит время выдавать результат, – и понимаешь, что всю его работу придется переделывать. И потом ты еще должен страдать, поддерживая его ужасные конструкции.

Как только мы избавились от него и еще от пары лишних людей, остались те, кого действительно было за что уважать. У каждого была собственная область работы, где он был компетентен и даже талантлив. Никто не был настолько туп, чтобы это вас угнетало. Избавившись от паршивых овец, мы тут же стали чувствовать себя виноватыми, как будто у нас было мало работы, потому что вокруг все усердно трудились, вот бы и нам так.

Стэн: В то время случилась известная история, которую Админ и сейчас постоянно поминает. В самом начале проекта Candygram Админ собрал совещание, чтобы поговорить о коде и стандартах, которым все должны следовать. Люди приводили свои аргументы за и против, потом Админ принял решение, и стандарты были выработаны. На повестке дня было несколько пунктов – семь или восемь. Мы с Марком голосовали за верные решения, а Монах Тук был против практически всех. На собрании Админ принял все решения в нашу пользу, потому что они соответствовали Совершенному Коду, и никто не мог бы поспорить с ними, разве что Монах Тук. Тук оценивал каждый метод в основном исходя из того, много ли нужно будет печатать: это был его главный мотив. Он был фанатом коротких синтаксических конструкций, которые требовали поменьше ударов по клавишам.

И вот это совещание, где мы говорим о том, какой подход будем применять, какими должны быть стандарты. Админ всегда уделял большое внимание стандартам кода. Если у него и был пунктик, так именно

здесь: он всегда терпеть не мог тех, кто делает что-то восемью разными способами, что усложняет поддержку кода. Мы разбирались, наверное, часа два. Наконец, Админ спросил, хочет ли кто-то еще что-нибудь добавить. Тук встает и говорит: «Мне не нравится запах в туалете. И не нравится освежитель. Он ягодный, а я этого не люблю». Админ побагровел и сказал: «Слушай, мы купили его сами. Другого в магазине не было. Так, о чем-то *серьезном* кто-нибудь еще хочет высказаться?» В этом был весь Тук.

Эндрю: **И вы от него избавились?**

Марк: Вот тут-то и началось самое интересное.

Уровень CMM – ниже нуля, или «Процессы, на кой черт нам все эти процессы?!»

> Постановка задачи: Дать нашим клиентам серьезное преимущество с помощью технологических новинок, расширяющих их взаимоотношения с покупателями.

Если назвать наши первые методы разработки ПО примитивными, то это значит не сказать ничего. После нескольких месяцев медленного прогресса мы достигли примерно того же уровня технологии, что и клан обезьян в начале кубриковского фильма «Космическая одиссея 2001 года», – с такими же дикими криками и ударами по черепу. К плюсам относилось то, что я уже узрел свет – дьявольский – власти и славу управления источниками. Для нашего кода на Visual Basic C мы самого начала мы применяли управление источниками. Тем не менее, большая часть нашей бизнес-логики – сотни тысяч строк на T-SQL – остались в заточении, то есть в нашей базе данных разработки. Наш процесс управления изменениями SQL начинался и заканчивался Админом. Все изменения проходили через него. Пришлось признать, что в его безумии есть метод. В конце концов, он был самопровозглашенным лучшим программистом на SQL в Питтсбурге (или на всем Восточном побережье?). Его оправдывало то, что он не был так уж самодоволен. Он часто рассказывал о разработчике и техническом колумнисте, что тот мог быть лучше, «но погиб, когда его снегоход врезался в дерево!..» И после своего фирменного приступа смеха, помрачнев, говорил: «Мне так жаль его семью: у него трое детей».

Мы даже не всегда знали, что должны делать. (Следующая фраза произносится голосом Джима Мора, главного тренера команды «Indianapolis Colts».) «Детализированные спецификации!? Детализированные спецификации!!?» Нам повезло еще, что у нас были детализированные

беседы. Админ всегда был в движении и вечно спешил. Человеческое общение было слишком ограниченным, чтобы заинтересовать его четверти головного мозга. Если у него было время, то он успевал отправить электронное письмо, где описывал, что мы должны делать. Если же он позвонил – считай, повезло. Не я один видел многие недостатки в наших процессах и жаловался на них, но никто не собирался предпринимать односторонних действий. Предполагалось, что большинство членов команды работают над Edsel – системой управления совместной разведкой нефти, но большая часть проекта этого приложения опиралась на функциональность приложения AS/400, которое никто из нас не видел и не имел к нему доступа.

Втайне я еще надеялся на некий план. Может, мы день и ночь корпим над задачей, как союзники в Блетчли-парке, взламывавшие во Вторую мировую шифр немецкой «Энигмы», – каждый сосредоточивается на крохотной части целого, никогда не видя достаточного фрагмента, чтобы понять, что именно мы делаем. И однажды последний элемент пазла ляжет на место, и тут явится провиденное Хедли и Админом произведение искусства – наша многофункциональная система учета, которую мы назовем Candygram (в честь особых конфет из «Сверкающих седел»[1]). Мы усилим техническую архитектуру, созданную для Edsel, и Rock Ridge вступит в дивный новый мир расстроенного энергетического рынка. Впрочем, нам скорее предстояло, как и прочим компаниям, сражаться за место под солнцем на перспективном рынке; наши попытки нередко проваливались.

Но вскоре в атмосфере забрезжили некоторые причины для оптимизма (собственно, благодаря нашему новому офису появилась и сама атмосфера). Стэн оказался удивительно одаренным сотрудником. И главное – его личность была даже важнее физического присутствия, к тому же он умел программировать. Команда оценила это, и до конца его пребывания в Rock Ridge он стал ее центром притяжения. То есть его отсек превратился в своего рода социальный центр, что, как водится, сказалось на продуктивности. Теперь у команды появился шанс сплотиться, и с помощью некоторых других сотрудников она составила эффективный противовес Админу. Улучшившаяся динамика группы сгладила наиболее резкие углы Админа и позволила заполнить вакуум, возникавший, когда его время и присутствие требовались повсюду. Это имело некоторый успех: Админ начал нам доверять, передавая все больше полномочий. Он даже стал порой прислушиваться к другим мнениям, а не только к своему. Предстоял еще долгий путь, но наконец-то мы по крайней мере начали разрабатывать процессы, стандарты и заготовку

[1] «Сверкающие седла» («Blazing Saddles», 1974) – фильм Мела Брукса, пародия на вестерны. Неоднократно цитируется в этой главе. – *Прим. ред.*

для действительно впечатляющей системы учета. Мы прошли еще одну критическую точку. Таким образом, ядро команды сформировалось, когда наши роли еще не определились, и пока ни один важный сотрудник не мог уйти.

Первым заказчиком Candygram было небольшое подразделение крупного энергетического предприятия. Нам повезло с ВТК (Весьма Терпеливой Компанией) в качестве первого клиента. И не только потому, что вторая по величине сделка, которую когда-либо подписывал отдел продаж, был наш счет из бара, который менеджер по продажам случайно забыл закрыть во время слишком затянувшегося часа скидок. ВТК напоминала домашнюю, не-очень-яркую первую девушку. Она мирилась с нашими низкокачественными релизами, с нашей небрежностью, давая нам шанс отточить нашу технику на настоящем, живом клиенте и изменить ее к лучшему. Не думаю, что они рассчитывали на что-то большее.

Пришлось ждать еще год или больше, чтобы получить хоть как-то повторяемый процесс выпуска. Лучшее, что у нас было, – это пакетные файлы для автоматизации применения сценариев релиза к базе данных. Что именно относится к определенному релизу и как провести эти изменения, не повреждая данные, определялось исключительно вручную. Два разработчика независимо друг от друга обращались к одной кодовой базе, не гарантируя, что в результате получится одно и то же: конечный продукт варьировался в зависимости от того, кто его создал и на каком компьютере. В итоге компоненты и/или объекты базы данных часто отсутствовали или были неправильно сконфигурированы, что вызывало самые разнообразные ошибки в основном трафике.

Я впервые осознал, насколько терпимой может быть ВТК при срочном выпуске последнего релиза базы данных, когда я как раз собирался обновить клиентское приложение. Админ назначил на сборку пакета базы данных относительно нового сотрудника, Вздыхающего – умного, трудолюбивого и молодого парня. Но выглядел он так, словно тяготы жизни преждевременно старили его прямо у вас на глазах. Я был поражен, увидев несколько лет спустя его фотографию из колледжа: он был олицетворением жизненных сил, но теперь казалось, будто он сделан из воска и оставлен на жарком солнце. Админ уверял, что он работал без передышки. Я дал этому парню такое прозвище (у каждого из нас их было несколько), потому что как только он появлялся поблизости, вскоре доносился сдавленный вздох. Внешнее давление изменяло его только на 90%: не так уж плохо, если не думать о чем-то вроде безопасности в полете или, что ближе к теме, кода базы данных. Вздыхающий патологически плохо печатал. Его опечатки были настолько невероятными, что, казалось, он делал их нарочно. Работая с одним из его проектов, я создал с полдюжины диалоговых окон с функцией Colon Batch.

Я знал, что это некорректно, но ведь не мог же он настолько регулярно набирать «Colon» вместо «Clone», правда? Над этим можно посмеяться, но когда я начал обновлять базу данных Candygram для ВТК, мне было совсем не смешно. Вздыхающий нравился мне с самого начала, так что я недолго колебался, начиная апгрейд. Он был «протестирован». Через несколько секунд я получил исключение, а таблица базы данных лишилась всех данных. Пока администратор клиентской базы данных восстанавливал ее, я говорил с Вздыхающим по телефону. Найдя ошибочную строку, мы начали заново. Следующие шесть часов мы повторяли цикл «выполнение–ошибка–отладка–извинения–восстановление».

Что даже Админу было трудно преувеличить, в чем он был настоящим мастером, так это энциклопедические знания о системе Candygram. То, что у нас не было управления изменениями, спецификациями и процессом, означало, что на эти знания мы полагались почти что полностью. И эта ситуация сама себя продлевала. Он единственный знал все, то есть ему и приходилось во все вникать. Делегирование процесса выпуска в те ранние дни могло быть особенно болезненным. Компании остро нужны были деньги, так что мы преждевременно выпустили систему, полностью сознавая, что важная функциональность еще далеко не совершенна. Мы познали радость разработки и поддержки живой системы, критичной для бизнеса клиента. Хуже было то, что обилие вопросов качества, всплывавших ежедневно, заставляло одного из наших специалистов по SQL неделями бороться со вспышками на местах на пределе сил. Поскольку главным фактором было время, наша разработка часто шла прямо на живой системе. Множество неудачных решений, приводящих к финансовым проблемам и вызванных ими, каждый день угрожало свести на нет все наши усилия.

Я усвоил, что такое качество, еще до прихода в компанию, именно такого качества мы и добивались. Я знал о нелинейной зависимости расходов на продукт от сроков. Но пришлось еще узнать, какую фундаментальную значимость имеет решение этой дилеммы для компании. Она была более важна, чем любое решение, которое можно было принять по поводу технологии, кадров или стратегии. И это был не просто ряд проблем. Важно было восприятие, особенно первые впечатления. Если бы мы потеряли доверие одного из первых заказчиков, то никакими усилиями, переговорами или унижением не смогли бы завоевать его снова. Проблемы качества могли отравить наши отношения, привести к дурным отзывам, лишить нас перспективы будущих продаж, ослабить наши позиции в сделках, что вело к необоснованным обещаниям, добавляя стресса и так уже находящейся на взводе команде разработчиков. Хедли, казалось, этого не понимал. Хотя мы продержались достаточно долго, чтобы обрести гораздо больше опыта разработки, мы еще не были настолько компетентны, чтобы разорвать цикл.

Дата: 04/22/1999 04:59 PM
От кого: Чистоган *cdollar@rockridge.com*
Кому: персоналу Rock Ridge
Тема: Candygram System 3.0

Как известно, Rock Ridge получила заказ на разработку системы Candygram 3.0 (CS3) для розничной торговли ВТК. Вся наша организация усердно трудится, чтобы достичь цели, и я хотел бы сообщить, когда необходимо сдать проект.

Бета-дата для CS3 – на данный момент 5 мая 1999 года. Бета-версия будет включать запросы пользователя, команды, настройку и выполнение установок, исполнение сервисного цикла, биллинга и записи информации, поддержку функциональности таблиц. Мы также будем внедрять роли безопасности, разрешения и эффективные сообщения об ошибках. За последние две недели мы провели бета-тестирование утилиты Candygram System Import (CSI) для ВТК. CSI была написана для обмена данными между RSSC (системой ВТКР) и Candygram. CSI также будет обрабатывать загрузку данных в защищенные папки. Это тестирование очень полезно не только собственно для CSI, но и для того, чтобы не повторить тех же ошибок в CS3. ВТК планирует начать использовать CSI, загрузив 130 000 клиентов на период бета-тестирования. Кодирование CSI завершится 26 апреля 1999 года.

Нам нужно продолжать сосредоточенно работать над тем, чтобы успеть к этим датам. В принципе система выглядит очень хорошо, и я доволен сплоченностью команды. Скоро начнется этап внедрения и поддержки проекта, и эстафета от проектировщиков и программистов перейдет к другим сотрудникам.

Быть лучшими непросто. Но надо делать вид, что это просто.

Проблемы качества были как противопехотные мины, которые могли долго находиться в спящем состоянии. Взрывались они в самое неподходящее время. Проблемы с проектом Edsel – донором Candygram – догнали нас через год после его сдачи. Пришлось героически работать, чтобы сдержать завышенные обещания крупнейшему нашему клиенту – компании-мультимиллиардеру, которая долгие годы вела бесчисленные войны на выживание с компаниями гораздо крупнее нашей. У нас же были только красные чернила, древняя система учета и предстоящие судебные иски. И это – рецепт успеха?

Дата: 09/29/1999 08:31 AM

От кого: Хедли Ламарр

Кому: Чистоган <*cdollar@rockridge.com*>, Стэн Гранайт *sgranite@rockridge.com*

Тема: обучение и внедрение Edsel

Теперь немного о Джей-Джее, Нытике и разработке Edsel. Они очень опечалены тем, что над системой пришлось работать так долго. Собственно, ее год делали для фирмы Sisyphus, пока те не отказались от ее использования. Пару недель назад Sisyphus решила все-таки ее использовать, дав нам на все неделю. Руководство фирмы знает, что не все в системе доделано до конца. Нам придется ограничить функциональность определением суммы комиссии и запросом проверки форм. Также мы будем переносить данные из AS/400 в Akron. В общем, предвижу, что на это понадобится несколько месяцев.

P. S. Джей-Джей может быть агрессивным и жестким. Он таков, но свое дело знает. Думаю, в принципе ему лучше давать, что он просит. Нытик же, наоборот. Не совсем в ладах с базами данных. Я бы предложил никаких изменений Нытика не внедрять, пока их не опробует Джей-Джей.

Бурая дыра, или «Думаю, все у вас есть»

Центр разработки программ (ЦРП), изначально был далек от идеального офисного помещения. Но надо сказать, что длительная дружба скорее завязывается в крысиной норе, чем в угловом кабинете. С первого взгляда можно было сказать, что мы теперь будем работать в чистом, хорошо освещенном, просторном офисе, даже с небольшой столовой. Естественный свет лил в окна, выходившие на главную и боковые улицы. Мы располагались на третьем этаже, под которым очень удачно находился магазин пончиков Брюггера. Парковка была на улице, но бесплатная, к тому же ее легко можно было найти среди окружающих зданий – явное преимущество по сравнению с деловой частью города. Рядом был Fat Head – наш любимый бар и официальный поставщик жареных крылышек для команды разработчиков Rock Ridge. Что может быть лучше?

Через несколько месяцев после переезда офис навсегда изменило еще одно потрясающее событие. Все начиналось как обычно: Админ по-прежнему фиглярствовал за столом для собраний. Это значило, что любой, кто находился в 50 футах от него, должен был участвовать в собрании, хотелось ему того или нет. Не помню, с чего все началось, но Джей-Пакс, видимо, решил, что Админ его достал. Казалось, весь офис притих, когда я услышал: «Пошел ты, Админ... Я устал от всего этого дерьма... держи свою fucking ручку». Может, я неточно цитирую, но четко помню, что впервые в офисе прозвучало слово fuck. Джей-Пакс, судя по всему, открыл ящик Пандоры, из которого стали вылетать ругательства. И стиль диалогов в офисе существенно изменился.

Идиотский вариант с единственным телефоном на офис, расположенным в центре, но неудобно, оказался интересным социологическим

экспериментом. Когда он звонил, все поднимали головы над перегородками, как койоты, в ожидании, кому же придется брать трубку. Глаза бегали туда-сюда, следовали обманные движения головой и фальстарты, пока наконец кто-нибудь не решался. Казалось, кто-то прочел книгу Тома Демарко «Человеческий фактор» и сделал все наоборот, чтобы создать идеальные условия для падения продуктивности. Если телефон и не нарушал «течение работы» (состояние повышенной продуктивности, которое наступает примерно через 45 минут после ничем не прерываемого труда) почти всех сотрудников офиса, он служил отличным рассадником простуды или гриппа.

Еще несколько минут поизучав офис 50 на 50 футов, вы быстро приходили к выводу, что примерно 20 отсеков с перегородками в половину человеческого роста плотно примыкают друг к другу и вряд ли способны обеспечить полную тишину и покой, когда все на месте. Кто, спрашивается, собирался сделать два больших отсека – каждый в половину офиса? И этот единственный беспроводной телефон в самом центре мог быть источником серьезных проблем. Затем вы, наконец, обращали внимание на две двери в дальней стене. Одна была кладовой, а другая *собственно* туалетом – без какого-то вестибюля или коридора. Один туалет размером с кладовку на весь офис. Двадцать человек живут на диете из острых крыльев и кофе – и один туалет. Это могло кончиться ужасно.

Эндрю: **Я пока не очень понимаю. О туалете действительно нужно говорить отдельно?**

Марк: Это была квинтэссенция работы в Rock Ridge.

Эндрю: **Серьезно? И когда вы пришли в компанию, все так и было?**

Марк: Нет. На самом деле в первом офисе, где дресс-кодом были рубашка и галстук, все было нормально: обычный скучный офис в деловой части. Так хотел Хедли. И первый парень, с которым я бы собрался подраться или повздорить, точно был бы не Стэн: его мощная фигура заполняла весь отсек.

В то время мы нечасто видели Админа, а другие персонажи, которые работали в офисе в деловой части, были такими скучными, что с ними и знаться не хотелось. Когда мы переехали на юг, Админ стал проводить с нами все больше времени, и мы наконец избавились от офисных лентяев.

Стало расти напряжение. Админ был весьма самолюбив, как, впрочем, и я, и Стэн. А когда вы о чем-то говорите на собрании, нужно выпустить пар. Высмеивать других значило убеждаться, что все мы в одинаковом

положении. Админ любил показывать, что его должность относительно важна – по сравнению с нами, прочими. Но мы всегда могли держаться с ним на одном уровне, переходя на личности.

Стэн: Админ начал обращаться с нами как с людьми, только когда мы стали его подкалывать.

Эндрю: **Но это была не просто пикировка. Судя по всему, у вас, ребята, насмешки доходили до невероятной степени. Многих довело бы до слез, окажись они в среде обитания сотрудников Rock Ridge. Как это вышло? И при чем здесь все-таки туалет?**

Марк: Туалет был благодатной почвой для шуток. Он был как актер, исполняющий сразу все роли, потому мы на нем и останавливаемся.

Мы писали код и выполняли остальную нашу работу, стараясь поскорее от этого отвязаться и вернуться к подшучиванию друг над другом. На собраниях преобладал примерно такой подход: «Так, теперь остановимся на минутку и немного попишем код, а потом вернемся к тому, чтобы... *(вырезано цензурой)*». И такое отношение к делу работало. Когда кто-то говорил, всегда можно было ожидать от слушателей новой идеи – как по коду, так и по личности оратора. И это помогало эффективности собраний. Они были исключительно продуктивными, как мне кажется, несмотря на постоянные перебивания. Никто на собраниях не говорил только потому, что хотелось поговорить. А если и хотелось, то говорили не о продукте, а о том, как у кого-то... *вырезано цензурой*. Думаю, это поддерживало сосредоточенность на работе. Работа выходила на передний план только тогда, когда идея была настолько хороша, что ее можно было обсуждать без шуток.

Стэн: Обычно нам нравились собрания, что, наверное, нехарактерно для всех остальных существующих в мире компаний. Мы любили встречаться друг с другом и говорить, потому что это было весело, так что никто не избегал собраний или не показывал виду, что ему неинтересно.

Марк: Никто даже не смотрел на собраниях в ноутбук.

Стэн: Ага.

Еще о сортире можно сказать, что это был своего рода Розеттский камень, лингва франка юмора. Если в комнате находилось 10 человек, то можно было не сомневаться: есть одна вещь, шутка о которой всем понравится, – дерьмо.

Эндрю: **Вот так и туалет вступил в игру.**

Марк: Да, здесь появляется туалет, потому что беспрерывный юмор действительно стал доминантой командной культуры. Мы только переехали в офис на юге и привыкали к тому факту, что двадцать сотруд-

ников пользуются одним и тем же туалетом прямо в середине офиса, в открытом пространстве без всяких стен или чего-то, что снизило бы эффект. Как будто в комнате находился огромный слон. Сортира просто *нельзя* было не заметить. И это словно сорвало какой-то клапан, и мы начали говорить обо всяких смешных вещах. Первое, в чем все мы были солидарны, так это то, что ситуация с туалетом была просто ужасной.

В любом другом месте работы туалет как-то скрыт. Там безопасно, и никто не будет обсуждать то, как ты туда идешь. В нашем же офисе он все время был у всех перед глазами. Мы видели дверь туалета, видели, как туда заходят, выходят; все знали, сколько там кто-то пробыл. Справиться с этим было никак нельзя, вы знали об этом постоянно – во всех смыслах слова. Так что сортир был вечной темой. Только представьте, какое важное влияние он оказывал на нашу совместную работу.

Начали с туалета, а потом стали шутить и по более тривиальным поводам. Но туалет продолжал лидировать в рейтинге острот, потому что дерьмо – одна из самых безопасных тем.

Стэн: Тут сработали два ключевых фактора, если отвлечься от туалета: во-первых, у нас не сложилась бы атмосфера товарищества, если бы нам не пришлось переезжать в этот инкубатор на юге. Подобная культура не могла возникнуть в старом офисе, похожем на формочку для льда, где коридорная дверь открывалась прямо в отсек Марка, где у всех был одинаково ужасный кофе, где мы были изолированы: ничего вокруг, никаких ресторанов, баров – ничего. Но потом мы переехали на юг Питтсбурга, когда разработчикам предоставили собственный офис в обычном районе города, где 24 часа в сутки что-то было вокруг; там было где посидеть вне работы и после нее. И там был туалет. Без всего этого такая культура не нашла бы выхода, и все было бы совсем не так.

Марк: На работе никто не отбывал срок. Никто не приходил, чтобы просто поприсутствовать. Все были на работе, потому что им нравилось. Либо это было забавно, либо нужно было сделать свою часть задач, чтобы забавно стало потом. Или можно было прийти, чтобы потом попасть на очень хороший обед или, может быть, ужин после работы. Ходить на работу было приятно. Иногда мы даже говорили: «Слушайте, я устал уже шутить. Давайте немного поработаем, чтобы перезагрузиться». Думаю, очень важно понять, насколько важны в таком подходе были южный офис и туалет. Активно работать нужно было только недолгий промежуток времени: краткий рывок, потом выход на обед, еще один рывок днем или вечером. Не надо было сидеть и пялиться на часы, потому что можно было уйти, когда вздумается.

Стэн: Тут крылись некоторые проблемы. Причем что бы вы ни делали – ржали над другими или занимались работой. Народ подобрался кру-

той, и нужно было не выходить из роли, чтобы по тебе не проехались катком во время общего подкалывания.

Но у нас были еще и нелепые цели проектов, над которыми тоже можно было смеяться. Если бы мы действительно делали все, что постоянно делают в «ответственных» конторах, где сначала планируют работу, а потом уже принимаются за проекты, которые считаются реалистичными, ничего бы не получилось. На нас налагались попросту абсурдные требования, и мы реагировали на них юмором висельника. Думаю, это помогло сплотиться против внешнего врага – идиотского расписания работы за не такие уж большие деньги; мы стали единым целым.

Марк: Каким-то образом Хедли всегда удавалось нас удивить. Хотя каждый раз повторялось все то же, мы почему-то все равно были в шоке.

Стэн: Напоминает историю лягушки, которую варят на медленном огне: «Что ж, ребята, я вижу, что к этому дедлайну вы успеваете, так что давайте его ужесточим еще на неделю».

Марк: Или уволить треть участников проекта, а дедлайн оставить таким же.

Стэн: «Давайте три месяца вести переговоры по контракту, но срок сдачи оставим изначальным». «Ну, думаю, вас не затруднит... у вас три недели, чтобы перевести проект на испанский». «У вас заработало? Так, а теперь пусть заработает в Пакистане, где потребителей в 25 раз больше, чем в самой крупной вашей предыдущей установке». И мы воспринимали это с каким-то беззаботным юмором, что почему-то помогало.

Эндрю: **Звучит так, будто вы преодолевали одно препятствие за другим – ужасное рабочее место, безумные дедлайны, увольнения – и каким-то образом ухитрялись стимулировать работу, хотя совершенно не таким образом, как учат в бизнес-школах.**

Марк: Это точно. В «Человеческом факторе» не сказано, что нужно воткнуть посередине офиса вонючий сортир, чтобы сплотить команду.

Ежедневные отходы в наших скудных удобствах вкупе с демографическими особенностями офиса повлекли за собой множество дискуссий скатологической тематики. Были среди них даже метаскатологические! Самая серьезная, после затяжного программирования, привела даже к прорыву в теоретической скатологии. Думаю, открытие вызвала Джей-Дог, наш технический менеджер и единственная женщина, способная вынести самые эксцентричные темы из обсуждаемых. Както она сердито заметила, что если в разговоре принимает участие достаточно парней, то рано или поздно всплывет тема дерьма. Я мечтал когда-то стать физиком-теоретиком, пока мой руководитель в Карнеги-Меллоне не заметил, что отсутствие у меня академической строгости

делает для меня предпочтительной карьеру компьютерщика. Воспользовавшись Стэном как экраном, я заметил поразительное сходство между указанным феноменом и следствиями пределов Чандрасекара и Толмана–Оппенгеймера–Волкова в астрофизике. Предел ТОВ – это верхняя планка массы нейтронной звезды, за пределами которой сила притяжения вызовет давление нейтронного вырождения и последующее преобразование звезды в черную дыру. Сходным образом я предположил, что любой разговор, в котором участвует достаточно мужчин, вскоре вызовет взрыв под действием собственной массы, сформировав сверхплотную дискуссию, в которой не смогут выжить никакие нескатологические дискуссии, и назвал ее бурой дырой. Теория хорошо уживалась с известными по наблюдениям данными. Возможные улучшения состоят в том, что скорость взрыва, судя по всему, увеличивается под воздействием алкоголя. К тому же взрыв можно предотвратить, если в разговоре принимает участие достаточно женщин, но необходимые экспериментальные данные для поддержки этой гипотезы собрать не удалось.

В некоторых из этих конвертов фондовые опционы, или «Пора действовать, и действовать быстро»

После того как мы расселись по отсекам и начали обустраиваться, Хедли приехал посмотреть, как его подчиненные расположились на новом месте. Всегда оставаясь самим собой, он немедленно отметил, что многие из нас сместили мониторы от центра стола немного вправо или влево. Это улучшение было немедленно запрещено как не согласованное с начальством. Это должно было послужить для меня ясным сигналом к следующей рассылке резюме. Человек, который не был способен поступиться своим эстетическим чувством в пользу комфорта разработчиков и не понимал, как это отзовется на команде, никак не должен был стоять у руля. Яснее Хедли не мог дать понять, как к нам относится. Мы явно не были в его глазах высокообразованными, умелыми профессионалами, составляющими ядро компании. Админ сразу почуял возникшее в атмосфере офиса напряжение или просто некую враждебность в моем лице. Он быстро вывел Хедли из офиса и попытался сгладить ситуацию. И это сработало. Теперь я сосредоточился на четкой проблеме, которая, на мой взгляд, имела простое решение: «На хрен его». Я поставлю монитор туда, куда хочу. Но нужно было понять, что с Хедли будут проблемы. Я упустил знак, что пора покинуть этот корабль. Впрочем, вопрос стал чисто теоретическим, едва мы поняли, что вряд ли Хедли еще раз переступит порог этого офиса.

Обещание альтернативных рабочих расписаний тоже заставило себя долго ждать. И когда они вступили в силу, мы были разочарованы.

Заметки с нашего первого собрания в ЦРП:

> Альтернативное рабочее расписание – мы по-прежнему работаем над деталями альтернативного рабочего расписания. Однако уточняем, что свободный график не предполагается. Сотрудникам будет предложено несколько вариантов жесткого расписания, из которых они смогут выбрать подходящий.

Он якобы доверял нам судьбу компании, но не хотел доверить нам приходить на работу, когда мы считаем нужным. Еще один досадный момент: мне нужны часы, чтобы оценить вашу производительность, потому что я не могу/не хочу интересоваться тем, что вы делаете на самом деле. Хедли продолжал демонстрировать, что его интересуют бочки дегтя, а не ложки меда. Мы увидим это и через несколько лет, когда в компании будет пересмотрена схема бонусов. Тогда у фирмы были большие проблемы с деньгами; урезали премии и уволили всех сотрудников, которые считались недостаточно важными. Были обоснованные опасения, что люди начнут массово увольняться, а мы не могли позволить себе потерять даже одного сотрудника, потому что оставалось слишком мало персонала. Для тех, кто изъявлял желание пережить кризис, был объявлен бонус. Детали, как обычно, были довольно смутными, но если мы оставались и компания выкарабкивалась, то нам предлагалась премия. Я не рассчитывал и на цент из этого, но мы оценили жест. К сожалению, не могу припомнить, чтобы бонус потом вообще упоминался. Когда мы сумели выжить и компания даже получила некоторый доход, Хедли, казалось, не был этому рад. Он так долго откладывал и так мало об этом говорил, что, когда мы наконец получили чеки, почти все сочли, что нас просто обманули (впрочем, как и ожидалось). И все мы были крайне удивлены, когда на одном из собраний действительно раздали чеки. Но своими словами и жестами он как-то ухитрился сделать день неожиданного платежа невеселым. Тот цикл увольнений был лишь одним из многих. Но именно он навсегда изменил ЦРП и, возможно, стал тем событием, которое раскрыло потенциал ядра нашей команды разработчиков.

«В одних конвертах фондовые опционы, в других – рекомендательные письма. Прошу подождать и не вскрывать конверты до моего ухода». Возможно, Хедли сказал и еще что-то, например про наш любимый требуемый объем продаж, но все мы услышали только это. Вся компания (человек 35) была вызвана в ЦРП на собрание без объяснений. Нас разместили большим кругом в центре офиса, прежде чем были розданы конверты. Он вышел из комнаты, а мы остались, сжимая одинаковые конверты из манильской бумаги и глядя друг на друга в недоуменном

молчании. Глаза бегали взад-вперед в поисках ответа или в ожидании сигнала начать открывать конверты. Мой конверт, однако, меня мало интересовал; у меня были более отдаленные, но потенциально более серьезные заботы. А точнее – кто же всех нас все-таки подставил, а именно, кто подставил меня? Недавно меня представили менеджеру по разработке программ, и это означало, что мне, возможно, дадут больше денег и возможность занять один из двух отсеков с высокой стенкой. И, в конце концов, право удобно парковаться на единственной стоянке, принадлежавшей офису. Организационная диаграмма описывала иную реальность: теперь все разработчики должны были отчитываться передо мной, а я – перед Админом, вице-президентом по разработке. Ни я, ни команда не сомневались, что что-то должно измениться. Админ оставался главной фигурой системы Candygram, и все должны были отчитываться непосредственно ему. Так что, не имея никаких прав и обязанностей менеджера, я смутно ощущал, что знаю, кто прошел отбор. Собрание было накануне в главном здании. То, что на мне галстук, указывало, что я там был. Интересно, не подумал ли кто, что я имею какое-то отношение к увольнениям?

Конверты вскрыли, и вскоре тишина нарушилась криками. Без всякого воодушевления или сговора мы направились в ближайший кабак через дорогу. Весь день мы плакались друг другу, кто больше, кто меньше. Сочувствие никогда не было моей сильной стороной, так что я остался с теми, кто вышел из испытания невредимыми. Слепой Страж относительно недавно присоединился к нашим специалистам по SQL. Это был мощный, рослый, с тяжелой челюстью парень из угольного штата. Добрый и внимательный, он, кажется, в прошлом был бойскаутом. Увидев нашего менеджера по качеству в одиночестве и явно расстроенной, он смело подошел, чтобы ее утешить. Она была не очень разговорчива. Но он продолжал рассказывать ей, как печально, что она потеряла всю команду, громко поинтересовался, что они будут делать дальше, как трудно будет теперь найти работу и как плохо, что изменения коснулись и ее. Она плакала все сильнее с каждым сказанным им словом. Наконец, она его остановила, сказав, что тоже уволена.

После этого Черного вторника обстановка в офисе круто изменилась. Почти все уцелевшие сотрудники были мужчинами, в среднем около 29 лет. Старшему работнику ЦРП было уже 34. Без «взрослых» офис быстро покатился по наклонной плоскости. За неделю офис стал напоминать одновременно фильмы «Зверинец» и «Повелитель мух», то есть дозволено было почти все. На каждую фразу мог последовать скабрезный ответ. До сих пор я стараюсь избегать определенных слов и фраз, за которыми тогда непременно следовала грязная реплика. (Например, недавно я в одной вполне культурной компании никак не мог решиться высказать простую мысль, что некий продукт животного происхожде-

ния в упаковке был на мой вкус пересолен.) Теперь порочные разговоры не ограничивались сортирной тематикой. Во время проверок проекта следовали такие комментарии, что от них покраснел бы даже актер Редд Фокс. Однако если проигнорировать всю эту пошлость, то получилась бы самая приличная и демократичная обстановка из всех офисов, в которых мне доводилось работать. Политика офиса, по крайней мере в применении к нам, прекратила свое существование (разве что все запасались хорошими закусками за тот короткий период, пока компания обеспечивала неплохой их выбор). Думаю, любой, кто не уважал бы нашу группу и не пользовался бы сам ее уважением, давно бы ушел. Но несмотря на жуткое количество работы, в него было приятно приходить каждый день.

В то время широкое распространение получила игра «Круги» (*http://www.urbandictionary.com/define.php?term=The+Circle+Game*), в которую мы играли не на подзатыльники, а на интерес. Она стала почти ежедневным ритуалом, и через несколько дней обычные попытки поймать кого-то в круг были признаны бесполезными. Ментальность холодной войны отразилась в том, что атаки и контратаки становились все более интересными и креативными. Через два месяца почти во всех подходящих местах красовались фотографии сотрудников в процессе игры. Такими местами могли оказаться холодильник, кухонные полки, подзеркальник и держатели туалетной бумаги, каталожные шкафы и даже компакт-диски. Один неустрашимый сотрудник (Слепой Страж) ухитрился между иннингами влезть на Джамботрон стадиона «PNC-парк», поразив более 20 000 зрителей. (Тут вы, конечно, скажете, что «Pirates» не собрали бы столько зрителей. Но тогда стадион только что открылся, и шли просто посмотреть новое место. К тому же в то время неудачи «Pirates» продолжались всего восемь сезонов и не все еще фанаты потеряли надежду.)

Теперь, думаю, вам понятно, почему работать почти всегда было весело, но, возможно, непонятно, как мы при этом ухитрялись еще что-то делать. В команде был почти нулевой процент брака. Если оставить в стороне пошлятину, общение было четким, быстрым и эффективным. Управление в традиционном смысле слова было практически излишне. Мы делали то, что нужно, постепенно приближаясь к созданию системы Candygram. Финансовую нестабильность компании мы простонапросто игнорировали. И каким-то образом еще и разрабатывали обычные системы отслеживания оттока денег. Товарищество и ритуалы, которые выработались у нас в этот период «Повелителя мух», сохранились даже после переезда нашего офиса на третьем этаже в чудесном районе Питтсбурга на 53-й этаж небоскреба на краю питтсбургского даунтауна (по-местному – дантана).

«Круги» тоже пережили переезд на 53-й этаж, хотя игра при незнакомых с правилами сотрудниках порой рождала ощущение неловкости. Как-то вечером несколько «правильных» сотрудников уже покинули офис, а несколько разработчиков пытались закончить собрание, пока не ушел Слепой Страж. Страж направился к лифту, продолжая разговор о проекте. Он решил, что коллеги и прибывший лифт представляют прекрасную, до сих пор не использованную, то есть оригинальную возможность сыграть в «Круг». Когда все зашли в лифт, он протянул руку к Полковнику (наш бухгалтер), чтобы создать круг. К несчастью, Полковник не знал игры и не был готов к тому, что крупный мужик, стоящий совсем рядом с ним, протянет к нему руку, причем примерно в направлении паха. Лицо Полковника, совершенно не представляющего, во что выльется эта поездка в лифте, отражало смесь шока, дурных предчувствий и омерзения; его физиономия навсегда запечатлелась в памяти нескольких сотрудников Rock Ridge.

Последовавший за этим окончательный переезд в пригородный офисный парк (что было вызвано более низкой арендной платой и близостью к дому Хедли) тоже не истребил ни игру, ни всеобщую увлеченность ею. Однажды все участники, кроме двоих, признали поражение и покинули поле боя, оставив двух разработчиков сражаться в легендарной суперсерии из семи матчей, пока один из них не применил умную жертву – чипсы со вкусом барбекю.

Хотя игра сохранилась, несмотря на переезды, той тесной связи, которая спаивала группу, выжить не удалось. Финансовые проблемы становились все ощутимее, и инженеров нередко посылали прямо к заказчикам, чтобы спасти буксующие проекты и сэкономить человекочасы. Это физическое разделение нарушило хрупкое равновесие. Лучшие шутки в духе «Harlem Globetrotters» теперь казались плоскими. Веселье исчезло, осталась работа – и притом не самая лучшая.

Блиц, или «Перекур закончен, мальчики, нечего тут валяться и загорать»

В начале 2000 года мы заключили свою самую крупную сделку – с Succubus Corp., крупной газовой компанией. На первом этапе нужно было внедрить имеющиеся функции. ВТК помогла с рекомендациями, но дело, подозреваю, решили слишком оптимистичные обещания насчет сроков и сметы второго этапа. Предполагалось, что больших денег мы не заработаем, но хотя бы расходы на разработку покроем. А потом уж начнем зашибать деньгу, по крайней мере так считалось. Несколькими неделями раньше мы праздновали «слияние и рост» нашей компании: мы собирались объединить усилия с интернет-компанией, которая

предоставила нам IP и финансировалась риэлторской компанией. Выданный по этому поводу кредит Хедли намеревался использовать на субаренду бывшей штаб-квартиры Alcoa на 53-м этаже второго по величине небоскреба города. Был приглашен мэр, который сказал несколько слов и, казалось, критически отнесся к нашим планам заработать 250 миллионов за несколько лет, так что на секунду приостановился, чтобы осмотреться, прежде чем огласить цифры. Впоследствии множество бестселлеров расскажут, как опасно переоценивать даже самые успешные небольшие компании. Но реальность никогда не учитывалась при планировании решений, так почему в этот раз все должно было измениться?

Нынешняя инкарнация системы Candygram могла работать с основными клиентскими функциями и счетами. На второй стадии мы договорились разработать куда более изощренные функции для учета и заботы о потребителе. Нужно было поддерживать много функций, необходимых для работы с газовыми службами, розничными газовыми и коммерческими энергетическими компаниями, транспортировкой газа и газопроводами. Параллельно с новыми разработками мы продолжали первый этап. Но и это было еще не все: у нас были демоверсии для крупных покупателей, в которые тоже нужны были новые функции. И это в дополнение к нашим постоянным обязанностям по поддержке текущих потребителей и системной разработке.

Осознав, что, удвоив или утроив нашу производительность, мы вряд ли улучшим результат, Хедли разработал смелый план: мы будем работать упорнее, намного упорнее.

> Дата: 10/31/2000 9:17 AM
>
> От кого: Чистоган *cdollar@rockridge.com*
>
> Кому: Разработчики, Качество, Техподдержка, Обучение и Технические писатели, Внедрение
>
> Тема: Ноябрьский блиц
>
> НОЯБРЬСКИЙ БЛИЦ в Rock Ridge
>
> Корпорация Rock Ridge в ноябре проводит четыре мероприятия: демонстрацию «Визжи как свинья»; презентацию для энергетических компаний «Слишком много для нас»; улучшение этапа I проекта Succubus (Товары и услуги); организацию этапа III проекта Succubus (Транспортировка и настройка). Вместе с этим будет продолжен этап II проекта Succubus (Газовая служба) – тестирование и написание, обучение, внедрение, разработка технической документации, онлайн-поддержка, программное обеспечение учебного курса и поддержка ВТК. Достижение этих целей докажет клиентам, инвесторам и нам самим, что мы – небольшая компания с большими возможностями.

> Я посчитал, что для достижения этих целей необходимо увеличить рабочий день. Чтобы сделать ноябрьскую работу более продуктивной (и чтобы вы при этом не сошли с ума и не потребовали развода), Rock Ridge оплатит следующее:
>
> - Отгулы
> - Ужины по будням
> - Обеды по выходным
> - Совместный ужин в ресторане в середине месяца
> - Корпоратив в конце месяца

В основном это касалось как раз команды разработчиков, поскольку нам предстояло решать главные задачи в плане этого проекта. Обучение, документация и внедрение могли быть в лучшем случае на подхвате. Спецификаций, позволяющих координировать усилия, не было. Приходилось соображать на ходу. В итоге всем пришлось долго бороться за то, чтобы просто быть в курсе новых функций, которые только что родила команда разработчиков.

Даже учитывая необходимость разработки нашего основного продукта, много усилий требовал один из самых впечатляющих наших трюков – создание программы, эквивалентной Turducken. Система Candygram (она же CS3), разработанная весной, была совершенно независимой от предыдущих поколений продукта. Но в одном из этих поколений был особый модуль учета газа, а в новой версии такого еще не было. Вероятно, заметив совпадение названий, руководство предложило нам перенести модуль учета газа в новую систему и уже пообещало, что на ближайшей выставке мы представим демоверсию именно этой концепции. Сначала разработчики высмеяли эту идею, но хозяин говорил серьезно, так что пришлось искать способы ее реализации. Стыдно признаться, но я нашел пакет запросов Windos API, благодаря которому окна скомпилированного Windows-приложения могли открываться как дочерние окна в нашей новой системе. Когда «главное» было сделано, оставалось только как-то разобраться с тем, что у систем были совершенно разные базы данных. Несколько сотен часов разработки – и преобразование данных позволило системе выглядеть убедительно, хотя она и почти не работала, что, впрочем, хозяев не беспокоило.

Требования к продажам, а затем и весь веб-сайт компании дали свежий повод для насмешек команды разработчиков. Вот пример – отрывок из отчета 2002 года фирмы, занимающейся исследованиями в энергетике:

> Candygram – самый известный продукт корпорации Rock Ridge, продвинутое программное решение, основанное на интернет-технологиях... В Rock Ridge утверждают, что функции программы обеспечивают возможность управлять всеми видами энергии: газом, электричеством,

нефтью, пропаном и другими товарами и услугами (вода, телефонная связь, системы безопасности, кабельное телевидение).

В 2002 г. система Candygram была клиент-серверной системой на базе Windows. Веб-версию Candygram, J2EE, Хедли еще на первом собрании по планированию продукта окрестил нарочито непонятным, но забавным именем Candygram: The Final Solution. Шесть лет спустя шутки по этому поводу продолжались, потому что все наши попытки применить Java™ разбивались об очередные проблемы с разработкой.

Заявление о том, что система Candygram имеет «возможность управлять всеми видами энергии», до сих пор заставляет меня громко хохотать. Так и вижу, как на презентации наш директор перечисляет все формы энергии, как Бубба в «Форрест Гампе»: «Газовая энергия, электроэнергия, тепловая энергия, кинетическая энергия, химическая энергия, ядерная энергия, вращательная энергия, потенциальная энергия, энергия притяжения, нервная энергия, сексуальная энергия...» Из-за этого заявления мне пришлось создать целый вагон регулярных выражений, чтобы перевести наш код основного газового модуля, написанный на VB и SQL, в модуль для любой энергии, как было заявлено на той неделе.

Но мы все еще надеялись, и все эти «шуточки» по поводу нашего офиса и нашей группы теперь усугублялись тем, что рабочая неделя отныне длилась по 80 и больше часов. Вечером или в выходные шутки не стихали. Они носились из угла в угол, каждый раз становясь все нелепее, чему очень способствовали недельные бессонницы.

Дата: 12/04/2000 10:47 AM

От кого: Чистоган *cdollar@rockridge.com*

Кому: Разработчики

Копия: Хедли Ламарр <*hlamarr@rockridge.com*>, Отдел кадров *t1000@rockridge.com*

Тема: Ноябрьский блиц

Ноябрьский блиц окончен.

Прежде всего я хотел бы поблагодарить каждого из вас за выдающиеся усилия. Мы не только успели к дедлайну, но и смогли с минимальными проблемами создать два новых модуля. Особо я благодарю Стэна Гранайта и Вздыхающего за создание архитектуры и проекта новых модулей. Это был новый опыт, и я надеюсь, что теперь модули стали лучше.

Во-вторых, мне известно, что большинству из вас приходилось проводить за компьютерами больше 80 часов в неделю. Теперь вы можете отдохнуть. Перезарядите мозги! У нас большие планы на первый квартал 2000 года, и нужно, чтобы все соображали ясно. У Марка Деновича есть таблица отпусков.

> В-третьих, тем, кто останется в офисе, пока другие будут отдыхать: мы не закончили с модулями Gas, Work Order и Transportation. Нужно многое доделать – отчеты, интерфейсы и еще по мелочи. Succubus очень доволен системой Candygram, и мы сейчас не можем снизить обороты. Наша команда внедрения убедится, что система Candygram работает как надо; вместе с разработчиками в декабре и ноябре они будут исправлять все возможные ошибки. Я бы хотел получить новый вариант системы Candygram к ближайшей пятнице.
>
> В-четвертых, планируется вечеринка по поводу окончания ноябрьского блица. Изначально мы собирались посетить игру «Penguins» или что-то вроде того. Но некоторые из вас говорили, что лучше бы поели мяса и сыра в офисе. Семеро из вас могут проголосовать, каким должно быть мероприятие (решать, впрочем, в итоге буду я). Пожалуйста, дайте мне знать о вашей договоренности как можно скорее.
>
> Еще раз благодарю за отличную работу!
>
> При составлении этого письма ни одно животное не пострадало и было использовано всего два восклицательных знака.

Мы не только усердно работали – мы усердно играли. Мы организовали все так, чтобы вечеринка за счет компании у меня дома совпала с играми чемпионата по американскому футболу. Мы заказали самую изысканную еду, какую только нашли в Интернете, столько элитного бухла, что У. К. Филдс от него бы помер, а заели это все 15 фунтами лучшего бифштекса, присланного из Чикаго.

Торжество, однако, оказалось несколько преждевременным. Несмотря ни на что, мы успели к назначенному сроку (или по крайней мере думали, что успели). Мы не знали одного – что пока мы выбивались из сил, наш заказчик тоже не терял даром время, передвигая ориентиры.

Приглашение на танец номер шесть, или «Не то чтобы вода и не то чтобы земля»

Кристофер Хоукинс, основатель и главный разработчик фирмы – производителя ПО, опубликовал статью «11 клиентов, с которыми нужно расстаться сразу же».[1] В ней Хоукинс рассказывает о типах поведения клиентов и заказчиков, которые считает агрессивными и каждый из которых считает достаточным поводом для разрыва деловых отношений. По моим подсчетам, фирма Succubus относилась по меньшей мере к семи типам, как я успел узнать более чем за три года ее взаимоотношений с Rock Ridge:

---

[1] *http://www.christopherhawkins.com/06-13-2005.htm*.

ЧТО-ТО НИ ЗА ЧТО: постоянно наращивает масштабы проекта, но отказывается платить за дополнительную работу.

ШАНТАЖИСТ: постоянно отказывается оплачивать счета, пока вы не выполните бесплатно дополнительную работу... подвид типа «что-то ни за что»... Шантаж – это всегда подход «выигрыш-проигрыш», а любой, кто руководствуется таким подходом в бизнесе, не стоит того, чтобы иметь с ним дело.

МЕДЛЕННЫЙ ПЛАТЕЛЬЩИК: постоянно задерживает оплату счетов.

КИДАЛА: постоянно выполняет свои обязанности с опозданием, при этом вас заставляет придерживаться графика.

ЛГУН: постоянно лжет вам.

ПРОВОКАТОР: постоянно выдвигает безумные требования, подвергая сомнению вашу профпригодность.

ДЕНЕЖНАЯ ЯМА: постоянно на мели... требует гораздо больше времени и усилий, чем те цены, которые вы можете им выставить... это может быть клиент, который требует постоянных скидок, дополнительной неоплачиваемой поддержки или же постоянно поступает так, что вам приходится работать упорнее.

До начала проекта Candygram фирма Succubus завалила уже два подобных проекта, что нужно отнести на счет трудностей, с которыми встретились их бизнес-партнеры и поставщики. Как хорошо организованная и долго существующее энергетическое предприятие она имела свою культуру, деформированную постоянными порочными стимулами, извлекаемыми из сочетания бизнеса, правительства и рабочей силы. Стандартными для них были такие процедуры, как конкурентные переговоры, максимизация выгод в рамках договора и сдерживание затрат. Большим влиянием в компании пользовался юридический отдел, который немедленно тащил партнера по бизнесу в суд.

С таким подходом бизнес вроде нашего, работающий неформально и не обладающий значительными финансовыми резервами, был полностью в их власти. К сожалению, мираж грядущих больших доходов означал, что наше руководство ради контракта готово было пойти на что угодно.

Началось все с МЕДЛЕННОГО ПЛАТЕЛЬЩИКА, который вскоре сменился сценарием ШАНТАЖИСТ, когда платеж был просрочен. Постоянное изменение требований влекло за собой значительные объемы неоплачиваемой работы, но мы согласились на это из-за обещаний потрясающих воображение выплат в конце. За этим последовал тип ЧТО-ТО НИ ЗА ЧТО: объемы росли, поскольку «прояснялись» критерии приемлемости.

Хорошо работала и тактика КИДАЛА. Переговоры по спецификации проекта продолжались еще шесть месяцев после желаемого начала са-

мого проекта. Единственной бесспорной датой был дедлайн, который так и не изменился. Важные чины в Succubus к тому же постоянно были недоступны, когда требовались. И их отсутствие было исключительно нашей проблемой.

Не могу утверждать, что они нам постоянно лгали. Но пример типа ЛГУН со стороны одного высокопоставленного члена команды проекта все же приведу. Можно предположить, что подобные закулисные инциденты были нередки, но в менее явной форме. В данном случае отличился ведущий аналитик-разработчик имеющейся мейнфрейм-системы; назовем его Р. По всеобщему мнению, Р. лучше всех в Succubus разбирался в имеющейся системе CIS. И не только по сумме знаний, но и потому, что Р. ревностно оберегал эту информацию, накапливая ее для своих нужд. Даже его работодатель был у него в руках. Если к системе хотели добавить функции или логику, нужно было убедить Р., что это стоит траты его времени. Р. понимал, что любая новая система ставит его положение по угрозу, и чтобы нейтрализовать эту угрозу, возглавил проект по созданию этой новой системы. Он стал судьей и присяжным, оценивающим новые функции Candygram. То, как далеко он был способен зайти, мы впервые узнали от нашего руководителя проекта Пи-Батта, который помогал нам с локальным тестированием. В тот день они тестировали довольно сложные финансовые расчеты, которыми можно было управлять с помощью нескольких настраиваемых параметров. Р. вызвал общее волнение, указав на ошибку в вычислениях при подключении одного из параметров конфигурации. Но он не знал, что Пи-Батт, сознавая всю важность этих вычислений, протестировал эту функцию сам, до представления ее в Succubus. И он потребовал предъявить проблему в ходе тестирования прямо здесь. Р. запустил программу и предоставил требуемые доказательства. Убедившись, что в его тестовой системе все работало, Пи-Бат создал новый тестовый сценарий в системе Succubus. Р. вновь запустил тест и опять получил данные с ошибкой, заявив, что программа не работает. Снедаемый предчувствием, Пи-Батт посмотрел в проверочные таблицы системы, где фиксировались изменения конфигурации. Эти таблицы были созданы для помощи в тестировании и поддержке у нас, а в объявленную функциональность системы не входили. Данные о пользователе и времени изменения свидетельствовали, что перед каждым тестом Р. отключал параметр конфигурации, включая его, как только получал подтверждение ошибки. Уличенный с помощью таблиц проверки Р. со злобой признался: «Да, наверное, я так и делал». Драки удалось избежать, но какой-то урон был нанесен. Позднее Пи-Батт пересказал эту историю, когда Р. подал в суд за неправомерный разрыв отношений.

Поведение типа ПРОВОКАТОР все более и более проявлялось по мере неотвратимого приближения даты сдачи проекта. Ближе к концу при-

мерно 40% наших сотрудников были вынуждены перерабатывать. Мы не очень протестовали, поскольку часы эти в принципе оплачивались, но из-за игр с платежными ведомостями мы так и не увидели этого столь необходимого нам вознаграждения.

Несомненно и то, что фирма Succubus была классическим клиентом типа ДЕНЕЖНАЯ ЯМА. Не имея достаточных резервов для компенсации затрат, мы дважды в ходе проекта оказывались на краю банкротства, и каждый раз в результате приходилось увольнять чуть ли не половину сотрудников.

> Дата: 11/20/2001 2:31PM
>
> От кого: Хедли Ламарр *hlamarr@rockridge.com*
>
> Кому: Персонал
>
> Тема: Проект улучшения системы Candygram для Succubus
>
> Нашей компании обещан денежный бонус в случае сдачи проекта до 25 января.
>
> Надеюсь, что если к дедлайну 7 декабря мы сдадим продукт без дефектов, у нас останется очень хороший шанс действительно получить некоторое количество бонусов.

Даже для Хедли уже было очевидно, что на оплату рассчитывать не приходится, особенно учитывая невероятно высокий барьер «ПО без дефектов». Но мы все еще ковыляли.

В конце июля 2004 года, не сумев заключить контракт с крупнейшей в Пакистане компанией природного газа (нереальный даже по самым оптимистичным меркам), мы оказались в самой большой финансовой яме за всю историю компании. Триумвират (Хедли, Пи-Батт и неохотно примкнувший Админ) решил подойти к проблеме с двух сторон: 1) немедленно начать новую серию увольнений; 2) предложить продать исходный код Succubus *и* отказаться от всех контрактов сотрудников Rock Ridge, чтобы проект продолжался только в рамках Succubus. Список сотрудников, обреченных на увольнение, отражал особенности мышления Хедли.

Среди них был единственный сотрудник, уделявший достаточно времени тестированию качества, и Стэн, разработчик клиентского ПО, который, возможно, лучше всех разбирался в системе и который шесть лет подряд, без преувеличений, работал сверхурочно. Внезапно из Succubus ему протянули руку помощи, выразив желание принять его на работу. Среди остающихся были два сотрудника, нанятые в последние шесть месяцев. Они не знали ни систему, ни наших текущих клиентов, но у них были именно те качества, которые нравились Хедли. С., аналитик результатов внедрения, выглядел и говорил как бизнесмен. Он носил широкие брюки и фирменные рубашки. Даже его манера речи нравилась Хедли: он про-

износил именно те слова, которые были в моде в то время. Резюме у него было убедительным, но поскольку он не мог давать результат в предельно сжатые сроки, его потенциал роли не играл. Другой сотрудник был еще хуже. Понг предявил права на звание «старшего разработчика», но незначительный кусок кода, который он написал, все же нуждался в переработке, хотя правильнее было его выбросить. Более того, он снижал и общую продуктивность, постоянно обращаясь к коллегам за легко находимой информацией. Чего у него было не отнять, так это способности уверенно нести ту первоклассную чушь, фанатом которой был Хедли. Как видите, у Хедли было собственное восприятие реальности.

Самые блестящие результаты принесла работа Понга над процедурами обеспечения качества в Rock Ridge, включенными в систему RFP. Дело в том, что все те незначительные процедуры по обеспечению качества, которые мы проводили, находились в ведении одного сотрудника-совместителя, не имевшего соответствующего опыта. Но согласно Понгу, мы явно лидировали в индустрии, причем выражался он настолько обтекаемо, что его слова не выглядели прямой ложью. Тут Хедли было чем поживиться.

Отрывок из RFP:

> Система обеспечения качества в Rock Ridge зиждется прежде всего на том, что почти весь наш технический персонал имеет дипломы с отличием по компьютерным наукам и/или программированию. Здесь чувствуется существенное влияние Университета Карнеги-Меллона и Института программирования, поскольку многие наши сотрудники – выпускники Карнеги-Меллона. В результате техническая часть нашей организации обладает почти универсальными знаниями в области Модели зрелости ПО и основанных на метриках процессов разработки многократно используемого ПО, ведущих свое начало от нее.

Перевод:

> Три представителя технического персонала имеют отношение к Карнеги-Меллону. Один ушел, причем не с компьютерного факультета, получив средний академический балл меньше единицы, хотя и до того, как его прямо попросили это сделать. Второго таки попросили, но потом он вернулся и почти закончил, хотя тоже не компьютерный факультет, а третий просто женат на бывшей сотруднице Института программирования, откуда она уволилась через несколько лет. При этом некоторые члены нашей команды усвоили Модель зрелости ПО – настолько, чтобы понять, что наши методы до смешного плохи, с какой точки зрения их ни оценивай, причем это несоответствие настолько вдохновило одного из нас, что он написал книгу об управлении программными проектами.

К концу лета 2004 года решение по коду и сотрудникам было уже почти признано окончательным. Хедли пожелал лично встретиться с каж-

дым разработчиком. Цель встречи зависела от вашей судьбы: если Хедли надеялся, что вы останетесь в Rock Ridge, то он пытался внушить вам свой взгляд на обновленную, полную сил компанию. Если вы стояли в списке Succubus, то встреча имела более сложную цель. Сделка базировалась на том, что хотя бы двоих из этого списка нужно было уговорить принять предложение Succubus. Поэтому ему отчаянно нужно было уговорить сотрудников дать согласие. Но годы общения с Succubus оставили неприятный осадок и у него самого. Он убеждал, но без особого пыла, в частности, говоря в свое оправдание, что в случае чего можно будет быстро уйти из Succubus. Однако он ухитрился без всякого согласования предоставить Succubus все записи отдела кадров, в том числе и о зарплате, что сильно ударило по сотрудникам, которые в ином случае могли рассчитывать на бóльшую зарплату.

Будучи одним из нанятых первыми сотрудников, я выбрал создание клиентской среды Candygram. К концу все уже были полностью загружены работой над проектом, то есть я снова мог выбирать. Редко сталкиваясь с бизнес-логикой, я не попадал в поле зрения Succubus. Это означало также, что я принимал участие – и порой больше, чем участие, – практически во всем, в том числе в почте и инфраструктуре сети. Так что меня не удивило, что разговор со мной был последним по времени и что Хедли потратил на меня свой основной деловой шарм. Мне предложили должность главного технического директора. За секунду-другую я испытал восторг и радость, затем сомнение и отвращение, пока не пришел, наконец, к отчаянию и обиде. Мне предлагали капитанскую фуражку, но на очевидным образом тонущем корабле. К тому же я понял, что инженером, упомянутым в примечании к бизнес-плану, был я.

> Rock Ridge направит специалиста в Гургаон (Индия) для обмена опытом. Предполагаемый срок командировки – 3–4 недели.

Мы собирались начать оффшорные разработки. Объектом одной из самых популярных наших шуток были наши возможные индийские сменщики. Меня не обижало то, что моих коллег заменят более дешевой рабочей силой. Я просто не мог понять, почему он после всех шестилетних уроков решил, что наше спасение – в неизвестной оффшорной команде сотрудников с почасовой оплатой, не имеющих прямого отношения к нашей компании, незнакомых с нашим продуктом и не преданных ему. Вот мое письмо после той встречи:

> Когда: Tue, 14 Sep 2004 10:48AM
>
> От: Марк Денович *mark@numbersix.com*
>
> Кому: Стэн Гранайт <stangranite@cablecompany.net>, Барристер <tubman@tubman.com>, РазмешатьБезКомков *buc@barestearns.com*

Тема: re: Днем встречаюсь с Хедли

Твою мать.

Марк

По крайней мере, он потратил на меня 30 минут. Поняв, что Хедли вряд ли выкроит время для индивидуальных встреч с ними (возможно, тут как-то сказалась его трехдневная поездка в Европу), Стэн, БЛ и Барристер предложили встретиться со всеми сразу. Он согласился, и это означало, что них будет уникальная возможность услышать из его уст оба варианта. Сначала Хедли начал прорабатывать вариант «останьтесь»: «БЛ, мы работаем с тобой почти десять лет. Мы с тобой уже долгое время и не хотели бы тебя отпускать. Хочу, чтобы ты знал: пока компанией руковожу я, у тебя всегда будет место в Rock Ridge». Потом он перешел к Барристеру: «Барристер, я очень ценю все, что ты сделал для компании. Проект Cash Drawer оказался очень полезным для нашей окупаемости, и я хочу, чтобы ты знал: если вариант с Succubus сорвется, у тебя есть куда отступать». Под конец Хедли собрался делать то же предложение Стэну. Но не закончив с Succubus, перешел прямо к делу: «Стэн, ты уволен с 31 августа». И тут же, без всякого перехода, завел свою стандартную речь, которую до того произносил тысячи раз: «Будущее светло и прекрасно, график продаж никогда не был столь оптимистичным, и если мы соберемся как команда...»

Это был конец эры.

Эпилог, или «Конкретно никуда... Всю жизнь мечтал туда съездить»

По правде говоря, финал наступил несколькими месяцами раньше – во время рокового обеда в Fat Head. К тому времени команда уже понесла серьезные потери. И тут у кого-то возникла светлая идея: пойти всем вместе пообедать.

Все было как в старые добрые времена, как будто мы снова оказались в ЦРП. Разговоры были быстрыми и жесткими. Остроумные ответы, опускания и пошлятина свирепствовали на прежнем уровне. Час обеденного перерыва превратился в два, потому что никому не хотелось уходить. Но вот стол опустел, последние напитки допиты, и мы с неохотой направились к двери. Погода снаружи была прекрасная, светило солнце, веял легкий ветерок. Мы попрощались. Я сделал шаг к машине и остановился. Мне в голову вдруг пришла мысль; как будто бриллиантовая пуля пронзила мой мозг. *«Господи, я опять иду на работу!»* РазмешатьБезКомков, мой приятель, которого я не так давно привел в компанию в отчаянной попытке помочь нам с вопросами процесса

и качества, заметил перемену в тот самый момент, когда она случилась. Между обедом и перспективой еще одного дня в Rock Ridge был резкий контраст. Мы держались вместе воспоминаниями. Офис больше не приносил удовольствия, и это, вероятно, случилось еще несколько лет назад. Мы просто этого не заметили. Возвращаясь, мы задержались немного, чтобы убедиться в сходстве своих ощущений. Мы ушли рано. Следующие несколько месяцев мы просто досиживали на работе.

Кое-кто из нас до сих пор работает на нашего великого клиента – укоренившись в недрах своих бывших мучителей, убеждается, что система Candygram до сих пор в строю. Многие перегруппировались для работы в компании, управляемой совершенно по-другому (главным образом прибыльной). И, как ни странно, некоторые все еще работают в Rock Ridge. Полагаю, продажи по-прежнему идут там полным ходом. Хедли, однако, пришлось продать компанию. По слухам, поживает он хорошо.

Эндрю: **Стэн, как я понимаю, вы ушли из Rock Ridge и полтора года не имели ничего общего с Candygram, но теперь вы вместе с Админом работаете в Succubus. Вы что, козел отпущения? Не те ли это странные отношения, когда вы знаете, что нужно уйти, но сладкими речами вас соблазняют остаться? Как вы оказались, так сказать, в четвертом кругу ада? Скажите, прежде всего, как вы устроились по окончании проекта.**

Стэн: Проще всего начать с того, чем мы занимаемся сейчас. Вы не поверите, но, работая в Succubus, мы с Админом не имеем никакого отношения к Candygram. В конце концов, код Rock Ridge был продан Succubus, и они, очевидно, изначально пригласили нас – меня и Админа – для поддержки Candygram. Думаю, нас ждали те же обязанности, что и в Rock Ridge, но мы с Админом даже не попытались применять наше знание системы.

Эндрю: **Даже несмотря на то, что это помогло бы обезопасить вашу нынешнюю работу?**

Стэн: Да. Мы решили дать людям возможность узнать все то же, что и мы. И оказалось, что они вообще понятия не имеют, как разрабатывать программы. Мы попытались ввести лучшее управление источниками, вообще лучшие процессы. Мы хотели помочь им в их карьере, научив их тому, чему сами научились в Rock Ridge методом проб и ошибок на всех наших проектах.

В общем, думаю, нам многое удалось, и я ушел в ту же компанию, куда поступил Марк после своего ухода из Rock Ridge. Еще один бывший со-

трудник Rock Ridge, Барристер, тоже там работал. Я провел там чуть больше полутора лет.

За это время Админ оставил команду Candygram. Поскольку мы оба работали над многими успешными проектами, они решили передвинуть Админа, а потом, когда я вернулся, и меня в главную ИТ-организацию, чтобы мы могли делать проекты и для всей Succubus. Этим мы сейчас и занимаемся.

Эндрю: **Итак, вы с Админом применяете лучшие методы из тех, что вы разработали — в основном в нечеловеческих условиях, при безумных помехах, с которыми имели дело в Rock Ridge, — и обучаете им представителей той же компании, которая и была одним из источников тех самых помех.**

Стэн: Именно так. То есть мы иногда делаем какие-то проекты с командой Candygram. Но Succubus меньше заинтересована в ИТ-ресурсах, чем в других сферах своей деятельности. Такие дела. По иронии судьбы, сейчас они привлекли к Candygram крупную консалтинговую компанию, которая оценивает все процессы, связанные с Candygram: достаточно ли денег тратится на поддержку системы?

Они пытаются урезать расходы: у руководства сложилось впечатление, что Candygram слишком дорогая игрушка. И оказалось, что консалтинговая фирма им сообщила: они тратят на систему меньше денег, чем их конкуренты – на свои. И теперь обо всем этом узнали другие.

Эндрю: **Судя по всему, от Candygram немало зависит то, как Succubus ведет дела.**

Стэн: Работая на мейнфрейм-системе, они не могли заглянуть в эту область бизнеса. Теперь, с Candygram, у них есть хранилище данных, куча отчетов и выборок данных, а также целая команда бизнес-аналитиков, которые всегда могут получить данные, необходимые для своих расчетов. Раньше этого не было.

Эндрю: **И все это основано на программе, которую вы разработали за несколько лет в Rock Ridge?**

Стэн: Да.

29

Стереотипы и нестандартные решения

Патрисия Энсворт

Эта история произошла в Нью-Йорке. Как и в большинстве нью-йоркских драм, она разворачивается вокруг недвижимости.

Главное действующее лицо – офисное здание в Нижнем Манхэттене, 12-этажная коробка без изысков, построенная вскоре после Второй мировой. Четверть века спустя выросшие неподалеку башни Всемирного торгового центра полностью закрыли его от солнечных лучей. Это здание – единственный подлинный, неизмененный и невыдуманный элемент этих псевдовоспоминаний. Уважаемые коллеги и адвокаты, прошу вас иметь в виду, что все прочее в ней (компания, люди, события, рассказчик) представляет собой набор фактов, которые, будучи реальными по отдельности, собранные вместе, составляют вымышленную картину.

Впервые это здание привлекло мое внимание, когда я проходила собеседование в компании «Фараон» на должность тестировщика ПО. Мне сразу понравился дух этого места. И хотя «Фараон» был издательством мирового уровня, специализировавшимся на финансовой аналитике и новостях, крупной и деятельной компанией с представительствами в разных странах, его штаб-квартира производила крайне умиротворяющее и дружелюбное впечатление. Внешне это здание не вызывало сильных эмоций, но интерьер был подобран со вкусом и любовью, фойе сверкало позолотой и мрамором, а люди тепло улыбались и здоровались в коридорах, словно все они были жителями маленького уютного городка.

Собеседование со мной проводил менеджер Джейкоб – худощавый, знающий, нервный человек, отвечавший за разработку систем форматирования печатной продукции издательства.

«Сейчас у нас в штате нет тестировщиков, – признался он. – В прошлом году мы взяли одного парня, но через несколько месяцев он решил податься в разработчики. По большей части тестирование производится нашими конечными пользователями и клиентами. Я не могу разглашать подробностей, пока ты не в команде, но они пропустили несколько очень серьезных ошибок, и это чуть не привело нас к банкротству. Нам нужен человек, который сможет подойти к делу более грамотно и системно».

Он взглянул на мое резюме: «Хм... у тебя степень магистра антропологии. Работала с программным обеспечением для раскопок или что-то в этом роде?»

Я объяснила, что изучала структуру и функции культурных систем с точки зрения коммерческой деятельности. В институте я подрабатывала написанием программной документации, а потом занялась тестированием программ. И обнаружила, что хорошо пишу контрольные примеры для тестирования на основании анализа того, как люди используют ПО в качестве инструмента связи со своими культурными системами. На тот момент я работала в международной финансовой компании и занималась тестированием удобства использования ПО, которое разрабатывалось в США, а применяться должно было по всему миру. Но вот недавно у нас была реорганизация, и мне не понравился новый начальник.

К моему удивлению и радости «Фараон» предложил мне работу. Я неплохо поладила с Джейкобом и с удовольствием думала о будущей работе с ним, но в пятницу (на работу я должна была выйти в понедельник) он позвонил мне и сообщил грустную новость.

«Мне очень жаль, но я ухожу из компании. Мне предложили потрясающее место в начинающей интернет-компании в Кремниевой долине. Сегодня я здесь последний день. Не волнуйся – у тебя все получится».

Утром в понедельник за мной в приемную спустилась элегантная, точеная темнокожая женщина примерно моих лет. Она представилась Барбарой, менеджером по проектам систем редактирования. Я проследовала за ней к лифту и на шестой этаж, где меня познакомили с менеджерами проектов по другим направлениям. Луиза, полная, по-матерински заботливая женщина, работавшая прежде менеджером по персоналу, теперь пестовала кадровые системы, сидя в зарослях любовно рассаженных ею растений. Менеджер проектов по всем внутренним и внешним приложениям на базе веб-технологий Кейтлин: узкая юбка и блузка с глубоким вырезом, шпильки, прекрасный маникюр и стильная стрижка; любопытно, подумала я, какими ветрами такую птицу занесло в мир программного обеспечения. Четвертый менеджер проектов Ник работал над производственными системами, осуществляв-

шими отправку публикуемых материалов в печать и на веб-сайты. Ник сообщил мне, что во время первой кампании в Персидском заливе служил в BBC, – этот факт подтверждала его военная выправка, опрятный внешний вид и неизменно корректное поведение.

Эта неофициальная «Большая четверка» руководителей проектов сообщила, что помимо систем, с которыми работали они, были еще несколько систем, команды которых обходились без менеджера проекта: финансовые системы, системы продаж и внутренней отчетности. В «Фараоне» не было четких процедур по управлению проектами или разработке ПО. Роль менеджера проекта сводилась к координации ресурсов, отслеживанию бюджета и составлению отчетов о выполненной работе. Настоящую власть представляло руководство уровня исполнительных директоров по разработке, которым подчинялись все разработчики и менеджеры проектов.

«Так кому буду подчиняться я?»

Повисло неловкое молчание.

«Меня принял Джейкоб, – продолжила я. – Он представился ведущим разработчиком софта для форматирования содержимого книг. Это по редакционной или производственной части?»

«Это производство, – ответил Ник. – Но у нас произошла реорганизация. Исполнительного директора, отвечавшего за внутреннюю отчетность, повысили до управляющего директора по разработке систем. Теперь все исполнительные директора подчиняются ему. Тебя уже познакомили с Дэйвом?»

«Возможно», – ответила я, пытаясь припомнить всех, кого встретила за три круга собеседований.

«Дэйв считает, что тестирование больше нужно для систем продаж, – сказала Барбара. – Но исполнительный директор, который курирует то направление, с этим не согласен. А менеджера проекта по этим системам у нас нет».

«Так кто будет *моим* руководителем?» – еще раз спросила я.

«Менеджер систем продаж, – ухмыльнулась Кейтлин. – Его зовут Скотт».

Чувствуя приближение приступа паники, я заставила себя несколько раз медленно и глубоко вдохнуть.

«Я правильно поняла? Мой новый начальник – менеджер направления, который ни разу меня не видел и который считает, что ему не нужен тестировщик ПО?»

Луиза похлопала меня по руке: «Это только временно. Все уладится, вот увидишь».

В сопровождении заботливой Барбары я прошла по всему офису и была представлена членам команды по разработке систем, а затем меня привели к кабинету моего нового начальника. Логово Скотта располагалось посреди длинного ряда одинаковых офисов, но отличалось от них, ибо в нем царили сумерки. Его обитатель убрал с потолка привычные лампы дневного света – их заменяли галогеновая лампа в стиле хай-тек на рабочем столе и лампа с матовым стеклянным абажуром на тумбочке. На стене, напротив фотографий подводного мира в рамках, висела индейская маска – тлинкиты или хайда[1]? – попыталась угадать я. Комната пропиталась табачным дымом.

Сам Скотт отличался от типичного разработчика ПО тем, что был одет с иголочки: брюки со стрелкой, деловая рубашка, галстук, запонки, начищенные до блеска туфли. В тот момент я решила, что он приоделся для встречи с высшим руководством, но вскоре выяснилось, что это его повседневный наряд. На вид ему было за тридцать, среднего телосложения, подтянутый, густые каштановые волосы до плеч, карие глаза и щербинка между передними зубами. Красавец, но не мой тип.

Я села в кресло через стол от него, положив левую руку на колено, чтоб было видно обручальное кольцо. Он открыл тумбочку и достал блокнот. Прежде чем он закрыл дверцу, я успела заметить внутри пачку сигарет и полупустую бутылку водки.

С минуту мы молча смотрели друг на друга. Наконец он обнажил зубы в некоем подобии улыбки, хотя глаза его оставались холодными.

«Значит, вы наш новый козел отпущения», – сказал он.

Начало было зловещим, но на самом деле потом все уладилось. После того как Скотт пару недель меня игнорировал в надежде, что я уйду сама, вмешался Дэйв (управляющий директор). И хотя я продолжала отвечать за тестирование систем продаж, он сделал моим прямым начальником менеджера проекта Ника, заявив Скотту, что тот должен изменить свое отношение.

Скотт ведал разработкой трех приложений по продажам: базы данных по управлению контентом для специалистов по продажам, базы данных по маркетинговой аналитике и системы исполнения заказов. Несколькими годами ранее он сам спроектировал и разработал все три эти системы, будучи еще начинающим разработчиком, на основании требований, подготовленных для него другом из отдела маркетинга. Приложения оказались очень успешными, того, кто инициировал тот проект, повысили до вице-президента по маркетингу и продажам, по-

[1] Тлинкиты, хайда – индейские народы в Канаде и США. – *Прим. перев.*

высили и Скотта. По каждому приложению теперь работали отдельный ведущий разработчик и пара младших разработчиков.

Однако Скотт слишком высоко шагнул. Он был великолепным разработчиком, но, взлетев до исполнительного директора, так и не захотел узнать больше о технологии разработки ПО или об управлении проектами. Множество ошибок и отказов в его системах было вызвано не плохим кодом, а неупорядоченностью самого процесса разработки. Его звездная репутация основывалась на том факте, что он готов был на все, лишь бы угодить заказчику, включая разработку уникальных версий приложений для отдельных пользователей и их собственноручную установку на компьютерах.

Я довольно быстро поняла, что Скотт встал на очень скользкую дорожку и боялся, что в любой момент лед под ним затрещит. Если бы он упал со своего высокого трона исполнительного директора, его богатая и известная семья, которая пришла в ужас от его решения играть в компьютеры, вместо того чтобы пойти учиться на юриста, смогла бы вдоволь позлорадствовать. Кроме того, все, кто с ним работал, знали, что он выпивает за обедом достаточно пива, а у себя в кабинете водки, чтобы можно было считать его человеком на грани алкоголизма.

В целом Скотт мне нравился. Я считала его умным, трудолюбивым, творческим и честным человеком. Ник побуждал меня заняться лечением технологической язвы, поразившей системы продаж. Но я никак не могла прорваться через выстроенную Скоттом систему обороны и развернуть мою стратегию тестирования или применить систему метрик, и мое терпение вскоре иссякло. Однажды, когда он в очередной раз вернулся с обеда навеселе, я пошла на таран.

«Скотт, мне хотелось бы убедить тебя, что мы по одну сторону баррикад, – начала я. – Я здесь не для того, чтобы шпионить за тобой по наущению Дэйва, критиковать или подрывать твое влияние. Я работаю на то, чтобы ты и твоя команда были на высоте. Как тренер или как редактор. Я действительно хочу вам помочь».

Возможно, Скотт почувствовал, что я была на грани увольнения, и его беспокоила реакция Дэйва на это, или, может быть, он почувствовал, что ничего не потеряет, если изменит свой курс. Как бы то ни было, с того дня мы перестали быть противниками и стали коллегами.

Забавно, но теперь и я прыгнула выше своей головы. Я умела разрабатывать и выполнять тесты и составлять по ним отчеты; но усовершенствование процессов разработки ПО и управления проектами требовали знаний, которыми я еще не располагала. Я накупила книг и по вечерам училась вместе с дочерью, которая тогда была в четвертом классе. Ник поддержал мой запрос об обучении, а Дэйв утвердил бюджет на

него. Мой муж взял на себя больше забот по дому и по уходу за ребенком, пока я ходила на курсы и участвовала в конференциях.

Несколько следующих месяцев мы медленно, но ощутимо двигались вперед. Тесно сотрудничая с восемью разработчиками систем продаж, Скотт и я занимались вопросами методики управления конфигурацией, управления версиями и изменениями и отслеживания ошибок. Это была нелегкая борьба, но в перерывах между схватками у нас было немало веселых минут. Постепенно мы научились уважать к мнение друг друга. Я чувствовала себя врачом, пациент которого, наконец, начинает самостоятельно выполнять необходимые упражнения, а улучшения проявлялись в снижении количества ошибок. Самые жаркие бои разворачивались вокруг вопросов замораживания кода и функций. Ник очень мне помогал в моем крестовом походе против «расползания» объемов работ и задач: он был тверд и непреклонен и обладал тем зрелым отношением к ежедневным кризисам, которое вырабатывается, наверное, у тех, кто уцелел в зоне настоящих боевых действий.

Интерес начали проявлять и другие исполнительные директора по разработке. Меня попросили помочь в найме и обучении тестировщиков для их систем. В трехмесячный срок у нас появилась Ирэн, которая занялась финансовыми системами (счета к оплате, счета к получению, выставление счетов, управление денежными средствами), в команду производственного отдела пришел Рэндалл (редакторская работа, производство, внутренняя отчетность, юридические и кадровые вопросы), а Фелипе взвалил на себя бремя веб-приложений. Менеджера проектов Луизу перевели из кадров в финансы, и Ирэн работала под ее началом. Рэндалл подчинялся Барбаре, а Фелипе – Кейтлин. Я же по-прежнему работала под руководством Ника. И хотя в организационных структурах менеджеры проектов и тестировщики ПО были распределены по группам разработки разных направлений, на деле капитаном команды менеджеров проектов была Барбара, а я стала капитаном команды тестировщиков.

Это было интересное и славное время. Новые тестировщики привнесли свежие идеи и подходы, а поскольку Ирэн и Фелипе пришли из известных компаний-разработчиков, их ожидания позволили поднять планку стандартов во всех группах. Дэйв позволил каждой группе разработчиков применять собственные процедуры, тем самым мы избежали многих конфликтов.

Само здание нашего офиса способствовало зарождению духа общности. У нас была столовая – еда, конечно, не высшего разряда, зато это было нейтральное и удобное место для общих встреч разного толка, от незапланированных собраний разработчиков до турниров в «Скрэббл» между командами. Фитнес-центр в подвале позволял нам физической

усталостью вытеснить любой стресс, а также перекинуться парой слов с конечными пользователями. Библиотека была благодатным убежищем от шума и толкотни в отсеках для любого, кому требовались уединение и покой для чтения, работы и размышлений. Еще одним популярным пристанищем стало просторное пустующее двухэтажное помещение склада на первом этаже, где раньше располагалось отделение банка. И хотя путь к нему преграждала тяжелая дверь на замке, все знали код этого замка, а также чтили неписанный кодекс, гласивший, что нельзя вмешиваться или сообщать о любых, даже запрещенных действиях, происходивших за этой дверью.

Да, дела шли хорошо, командный дух и продуктивность продолжали повышаться... Пока Скотта не бросила его девушка.

Внезапно Скотт стал мрачным и раздражительным, его обеды становились все продолжительнее, а возлияния обильнее. Мы с ним никогда не обсуждали личную жизнь, так что эту новость я узнала от Рэндалла, который подружился со Скоттом на тренировках в фитнес-центре. Ведущие разработчики приложений по продажам жаловались, что Скотт начал радикальный пересмотр разрабатываемых ими программ, как будто, снова погружаясь в программирование, он думал отвлечься от своих проблем.

Однажды в четверг вечером вся команда разработчиков систем была приглашена на вечеринку в знаменитый ресторан Windows on the World по случаю выпуска главной версии официального веб-сайта «Фараона». Было очень весело, и вид на залив в лучах заходящего солнца был великолепным, но мне пришлось уйти в разгаре празднования, чтобы отпустить няню и остаться с ребенком.

На следующий день Скотта не было, наша часть офиса была пугающе тихой, а вся группа выглядела усталой, раздраженной и подавленной. Полагая, что все дело в алкоголе и похмелье, я занялась своими делами и почистила папки в почте.

Но в понедельник кабинет Скотта снова был пуст – причем совсем пуст, чисто убран, и из него исчезли все личные вещи. Даже лампы дневного света вернулись на свое законное место.

Я нашла Анупама, ведущего разработчика приложений по маркетинговой аналитике, и спросила у него, что произошло. Он покачал головой, провел пальцем по горлу, пожал плечами и углубился в программирование.

В поисках Ника я оказалась в столовой, где он и Кейтлин как раз заканчивали завтракать. Не дожидаясь приглашения, я подсела к ним за столик.

«Привет! Кто-нибудь знает...»

«Ты о безвременном уходе Скотта?» – уточнил Ник.

«Именно».

Кейтлин засмеялась: «Ты пропустила большой скандал на вечеринке».

«Скотт напился как свинья, – сказал Ник. – Взял Дэйва в борцовский захват и начал его оскорблять».

«Да, пожалуй, физическое нападение на руководителя – серьезный повод для увольнения, – признала я. – У него были претензии к руководству Дэйва?»

Кейтлин засмеялась еще громче: «Не совсем так. Он назвал Дэйва голубым и заявил, что тот занимает свою должность только благодаря шашням с Бобом Уиллером».

«Уиллером? Главой по Латиноамериканскому региону? Но он же женат, да и Дэйв тоже».

Кейтлин перестала смеяться. Они с Ником устало переглянулись, а потом посмотрели на меня оба с сочувствием.

Я покраснела, чувствуя себя ребенком, который за взрослым столом опрокинул горящую свечку.

«Но это... этого не может быть», – я даже начала заикаться.

«Дальше все было еще лучше! – сказал Ник радостно. – Скотт не выпускал Дэйва, и тот начал вырубаться, тогда вступился Пол и хорошенько врезал Скотту, так что тот влетел в стол. Он смог сам отползти, но, я слышал, сломал пару ребер. У Дэйва растяжение шейных мышц».

«Ого. А я думала, что увлекательно провожу время, рисуя с дочерью плакат для ее научного проекта».

В тот день мне трудно было сконцентрироваться. Ник посоветовал мне просто работать как обычно, пока все не прояснится. Я провела запланированную встречу с Анупамом и двумя другими ведущими разработчиками систем продаж. А в конце рабочего дня мне позвонили из офиса Дэйва, срочно пригласив к руководителю.

Угловой кабинет, занимаемый Дэйвом, был раза в три больше, чем кабинеты исполнительных директоров. Я не бывала там прежде, и путь от двери мимо дивана и столика к креслам перед его столом показался мне бесконечным. Беглый осмотр комнаты выявил отсутствие семейных портретов и личных предметов, кроме фотографии гончей собаки в рамке. Дэйв жестом предложил мне сесть, и я присела, щурясь от яркого солнечного света, бившего мне прямо в глаза.

Дэйв был невысок ростом, он сидел в большом кожаном кресле, и на шее у него был ортопедический воротник. У него были тонкие руки, изящные черты лица и светлые прилизанные волосы. Я уже общалась с ним, но ни разу у нас не было серьезного или продолжительного разговора. Он заговорил таким низким и тихим голосом, что мне пришлось податься вперед, чтобы его расслышать.

«Ты знаешь, тестирование программного обеспечения относится к накладным расходам, – начал он с любезной улыбкой. – Иногда трудно доказать его пользу для бизнеса. Все начинания по улучшению качества всегда страдают первыми, когда урезают бюджет».

«Понимаю», – согласилась я. На самом деле, я была в замешательстве. Смысл сказанного полностью противоречил выражению его лица.

Судя по началу, меня собирались уволить. Но оказалось, наоборот, меня повышали. Дэйв сообщил, что хочет, чтобы я возглавила официальную группу по обеспечению качества. Подчиняться я буду непосредственно ему. Тестировщики будут на 50% подчиняться своим исполнительным директорам по разработке и на 50% мне.

Другие изменения в организации предполагали, что Анупам, разработчик отдела маркетинговой аналитики, заменит Скотта на посту менеджера по разработке систем продаж. На смену нам будут наняты новый менеджер проекта и тестировщик для систем продаж. Тем же порядком, что и в группе по обеспечению качества, Барбару повышали до уровня руководителя новой группы по управлению проектами. Менеджеры проектов на 50% подчинялись своим исполнительным директорам по разработке и на 50% Барбаре. Дэйв ожидал, что мы вместе с Барбарой будем внедрять усовершенствованные процедуры управления проектами и разработки ПО во всей группе.

Когда Дэйв описал мне мои новые функции, я почувствовала себя взволнованной и польщенной, но и несколько встревожилась. Я чувствовала, что работа под руководством Дэйва будет значительно отличаться от работы под началом Ника. У Дэйва не было никакого опыта работы в проектировании ПО. Обладая дипломом магистра гуманитарных наук, он был искушенным пользователем внутренних систем финансовой отчетности, и главный вице-президент по информационным технологиям решил, что у него хорошо получится возглавить подразделение разработки ПО. На позиции исполнительного директора по разработке он превзошел своих коллег в выполнении заданий в срок и в контроле расходов – но и в постоянной текучке кадров.

Это была серьезная реорганизация. Менеджеры по разработке не обрадовались идее разделить руководство тестировщиками и менеджерами проектов с другими руководителями. Наши бывшие коллеги неловко

себя чувствовали оттого, что им теперь нужно было подчиняться мне и Барбаре. Нас волновал вопрос структуры подчиненности, и мы не понимали, как договариваться с менеджерами по разработке о выстраивании приоритетов для проектных менеджеров и тестировщиков.

Среди новых обязательств, возложенных на меня и Барбару, было посещение раз в две недели совещаний у Дэйва, где все напрямую отчитывались перед ним о выполненной работе. Перед первым таким совещанием Дэйв попросил каждую из нас подготовить презентацию о наших целях и стратегиях. Мое волнение по поводу этого дебюта достигло апогея, у меня начинала кружиться голова и учащалось дыхание. Я знала, что собираюсь сказать, и тщательно отрепетировала эту речь перед заботливой семейной аудиторией, но сама мысль о том, что мне придется одновременно убеждать целую группу руководителей по разработке, вызывала у меня приступы паники и страха сцены.

На совещании я впервые сидела за круглым столом с исполнительными директорами, которых знала только понаслышке. Помимо Дэйва, его главы секретариата и Анупама присутствовали еще четверо новых коллег, с которыми мне предстояло отныне общаться ежедневно.

Исполнительный директор по финансовым системам Марк – гибкий, общительный и веселый разработчик на Visual Basic, яростный сторонник методики RUP. В свои 35 он выглядел лет на десять моложе. Женился сразу после университета на девушке, в которую был влюблен еще в школе, и уже успел стать отцом троих детей. Любимое времяпровождение вне работы – руководить бойскаутским отрядом своего сына. Когда бы и кто бы ни собирал средства на любые благотворительные цели – на Марка всегда можно было рассчитывать.

Рядом с ним мрачной скалой возвышался Пол – исполнительный директор по разработке веб-приложений, тот самый бугай, который сломал Скотту ребра. Крепкий, тучный, но мускулистый, бочка пороха с тлеющим фитилем, он жил один в берлоге, заваленной электронными музыкальными инструментами и компьютерными железками. Как-то, ненадолго оторвавшись от кодирования на C++, он случайно стал отцом внебрачного ребенка, мать которого теперь регулярно слала ему повестки в суд. В свободное время он гонял на своем Харлее по Адирондакскому парку. Кроме того, он был главным наркодилером нашей группы.

Напротив сидел Зви – исполнительный директор по производственным системам. Зви было за пятьдесят, и он работал в «Фараоне» с незапамятных времен. Это был ортодоксальный еврей, всегда одетый в бесформенный черный костюм и потертую белую рубашку с открытым воротом. За толстыми линзами очков в металлической оправе его глаза казались неестественно большими. Насколько мы могли знать, у Зви

не было никаких хобби, потому что он жил с женой и шестью детьми в округе Роклэнд, а это два часа дороги, в одну сторону. Зви мог просидеть, не шелохнувшись, невероятно долго и был замечательно внимательным слушателем. Весь персонал отдела разработки приходил за советом к Зви, если возникали технические или политические проблемы.

Четвертым человеком, которому меня официально представили, была Кэрол Чу. В отличие от других руководителей, которых называли просто по имени, исполнительного директора по разработке баз данных всегда называли полностью: «Кэрол Чу». Ростом Кэрол была не выше 140 сантиметров в кроссовках и почти всегда улыбалась. Она родилась и выросла в Шанхае, а потом стала доктором математических наук в Калифорнийском технологическом. У нее был муж – профессор физики, и десятилетний сын – чемпион по шахматам. Говорили, что она знает расположение каждого указателя в каждом индексе в терабайтах данных серверов «Фараона» за все время его существования. И хотя от нее частенько приходили сообщения со временем отправки от двух до пяти утра, она очень по-доброму относилась к своим подчиненным, как к большой семье, и обязательно отмечала день рожденья каждого.

После светской болтовни о провальном выступлении «Mets»[1] и о новом ресторанчике в нашем квартале Дэйв открыл собрание. Он поприветствовал новичков в команде – Анупама, Барбару и меня, затем секретарь раздал всем повестку совещания. Это был первый день Дэйва без ортопедического воротника, и он неубедительно пошутил насчет того, что нужно быть осторожным, «не высовываться».

Первым пунктом повестки была реорганизация команды. Дэйв попрежнему хотел, чтобы Ли, ведущий разработчик базы данных управления контактами, и Рэйчел, ведущий разработчик кадровых систем, поменялись ролями.

«Но у нас всего несколько недель до главного внедрения! – запротестовал Анупам. – Ли работала над этим релизом почти полгода».

«Рэйчел тоже, – сказал Зви. – Это ее первый релиз со времени ее повышения. Она очень волнуется и гордится своей работой. Ей будет очень неприятно уйти сейчас».

«Но в этом-то все и дело, – сказал Дэйв. – Мы не хотим, чтобы разработчики слишком лично относились к тому, что они программируют, или привыкали к конкретному сообществу пользователей. Наша группа разработчиков – единая команда, и мне бы хотелось, чтобы каждый игрок в ней был готов поменяться ролями по моему указанию».

---

[1] «New York Mets» – профессиональная бейсбольная команда Восточного дивизиона Национальной лиги США – *Прим. перев.*

В тишине, последовавшей за этими словами, я посмотрела на сидевших за столом людей. Если верить тому, что пишут о языке тела, исполнительным директорам не очень нравился управляющий директор, которому они подчинялись.

Моя презентация прошла гладко, хотя, если честно, похоже, никто к ней не прислушивался. После мне задали несколько вежливых вопросов, а Дэйв улыбнулся, поэтому я сочла выступление успешным и вздохнула с облегчением.

Но работа оказалась еще сложнее, чем я ожидала. Если при работе со Скоттом по систематизации рабочих процедур и системам продаж я чувствовала себя лечащим врачом, сейчас я ощутила себя скорее устроителем общественных мероприятий. Дэйв только платил мне зарплату, больше никакой помощи от него я не видела; не было ни поощрений, ни наказаний для исполнительных директоров, побуждающих их сотрудничать со мной для повышения качества. Очевидно, он ожидал, что я сама буду договариваться с каждой командой и разработаю стандарты и процедуры, с которыми все смогут согласиться.

Дело шло очень медленно, особенно когда исполнительные директора начинали бесконечные прения, отстаивая свои позиции. У тестировщиков, которые подчинялись мне, были собственные планы работ и ожидания. Однако дюйм за дюймом мы двигались вперед. Мы решили, что иметь единую документацию или методологию для всех приложений нецелесообразно. В конце концов мы разработали формы и перечни, в которых были обязательные и опциональные компоненты для определения требований, планирования тестов, их выполнения, отчетов по их результатам, отслеживания ошибок, управления изменениями, версиями и конфигурацией.

Барбара оказалась в очень похожей ситуации с группой по управлению проектами. В некотором отношении у нее было больше влияния, поскольку начальник Дэйва, главный вице-президент по информационным технологиям, а также влиятельные лица, от которых зависело финансирование проектов, хотели больше знать о разработках нашей группы. По сути, им хотелось лучше понимать, как, куда и почему тратятся их деньги. Первым делом Барбара ввела обязательный документ об объемах работ для всех новых проектов, а также процедуру проверки, определяющую финансирование, выделение ресурсов и планирование сроков. Когда это нововведение оказалось успешным, ввели необходимые формы и перечни проверок по структуре классификации работ, диаграмме Ганта, таблице распределения ролей и ответственностей, таблице рисков, интегрированному плану проекта и заключительному отчету.

Почти через год и Анупама, и Барбару, и меня еще раз повысили. Анупам стал исполнительным директором, а Барбара и я стали директорами. У меня появились акции компании и отдельный кабинет.

Мы с Барбарой отметили повышение походом по магазинам. Было начало сезона праздничных распродаж, и я обнаружила подземный переход из подвала нашего здания в вестибюль Всемирного торгового центра. Нам нравились торговые галереи Всемирного центра из-за их низких потолков, создававших уютную атмосферу. Там были всевозможные магазины, от самых простеньких (ремонт обуви, фотопечать, аптеки) до роскошных (одежда, электроника, ювелирные украшения). Самое удобное, что там можно было погулять в плохую погоду. Я купила себе маленькое черное платье. Барбара – изысканные сапоги. И обе мы купили украшения для своих дочерей.

Экономика процветала. Бизнес «Фараона» стремительно расширялся. В середине февраля, после годовой премии Дэйв объявил, что в целях обеспечения все возрастающих требований к ПО мы будем отдавать часть задач в аутсорсинг. Согласно плану через год наша группа должна была на треть состоять из разработчиков и тестировщиков из другой организации, работающих по контракту. Первоначально эти люди должны были находиться у нас, в здании штаб-квартиры «Фараона», но в итоге эти ресурсы могут располагаться в собственных помещениях своих организаций в любом месте США или за границей. Никто из сотрудников «Фараона» не должен был потерять работу.

За этим последовал интенсивный раунд презентаций аутсорсинговых компаний, стратегических совещаний по ресурсам и консультаций. Возникало ощущение, что очень многие важные решения принимались специалистами по снабжению и юристами, которые слабо разбирались в разработке ПО. Анупам активно лоббировал одну индийскую компанию, но в конце концов наш главный вице-президент по информационным технологиям выбрал компанию Quogue Consulting, основываясь на обзорах Gartner & Forrester[1] и обсуждениях со знакомыми из его делового круга.

На совещаниях у Дэйва появилось новое лицо. Фрэнк, менеджер по услугам Quogue, стриженый ежиком брюнет за сорок с обветренным лицом и дергающимся подбородком. У него были накачанные руки, плечи и грудь профессионального тяжелоатлета. На совещаниях он сидел напряженный и сжатый, как пружина. Мы выяснили, что Фрэнк работал в Quogue пять лет. До этого он служил на флоте, работал управляющим производством на каучуковой плантации в Малайзии и руководил пер-

[1] Gartner & Forrester – ведущие мировые аналитики рынка информационных технологий. – *Прим. перев.*

соналом круизного лайнера. Когда совещание закончилось и все встали из-за стола, я увидела у него на шее медальон в виде горгульи.

На выходе он перехватил меня и Барбару.

«Леди!» – он улыбался, но тон, которым он это сказал, больше пристал моряку в борделе, чем джентльмену на приеме.

«Добро пожаловать, – сказала я. – Мы рады новому сотрудничеству».

«Хотел сказать вам, что в предыдущем проекте моя команда получила сертификат соответствия 3-му уровню CMM[1]. Если вам интересно, буду рад поделиться с вами моим опытом».

Мне стало не по себе от высокомерия и покровительственного тона, которым это было сказано, но Барбара отреагировала иначе.

«Замечательно! Спасибо. Уверена, мы многому сможем у вас научиться», – расцвела она.

Вторжение Quogue на нашу территорию началось сразу же. Штатных тестировщиков, работавших под моим началом, поставили на приложения, где велась основная часть новых разработок, а люди, пришедшие по договору, должны были работать с изменениями, вносимыми в более стабильные старые системы. Мой тестировщик систем продаж уволился, чтобы продолжить учебу, и Фрэнк поставил на его место Рафика. Алекс начал тестировать системы счетов к оплате и к получению. В команду производственных систем пришла Наталья.

Мое участие в приеме новых тестировщиков сводилось к нулю. Соответственно, и исполнительные директора и ведущие разработчики не участвовали в найме программистов в Quogue. Следуя «модели увеличения численности персонала», Фрэнк требовал от нас только должностных инструкций, и сам приводил в команду все новых и новых людей. Он настаивал и на том, чтобы управлять рабочими процессами привлеченных сотрудников, поэтому поначалу посещал все совещания команды разработчиков и получал копии всех электронных сообщений, в которых его людям выдавались задания. Частично это объяснялось тем, что у Quogue были собственные процедуры отслеживания выполняемой работы, но кроме того казалось, что он искал возможность указать Дэйву на нашу несостоятельность, чтобы разрекламировать собственную компетентность и еще больше увеличить объем работ, выполняемых Quogue. Особенно Фрэнк раздражал ведущих разработчиков, они пытались сопротивляться его навязчивому поведению контролера их

[1] CMM (Capability Maturity Model) – модель зрелости процессов разработки ПО, разработанная Software Engeneering Institute (SEI) и применяемая в ИТ-индустрии для оценки процессов разработки в компаниях; всего по данной модели определены 5 уровней. – *Прим. перев.*

деятельности по интеграции привлеченных сотрудников в их проекты и команды.

Тем временем бизнес «Фараона» продолжал расширяться. Вскоре стало очевидным, что архитектура управления данными, внедренная, когда компания была в три раза меньше, уже недостаточна для нынешнего уровня и могла поставить под угрозу будущее расширение. Всем нам крайне не нравилась идея операции на главном нерве ИТ-систем, но мы понимали необходимость проектирования нового центрального хранилища данных и изменения всех входящих и выходящих систем. Одна мысль об этом бросала всех исполнительных директоров в жар. На наших первых совещаниях по сбору требований и планированию тестирования настроения переходили от истерических к упаднических, люди спешно обновляли свои резюме.

Положительным для меня было то, что у меня появлялась возможность возглавить крупный ответственный проект по тестированию ПО, в котором были задействованы разные службы, и я была взволнована этой сложной и интересной задачей. В целях обеспечения соответствия нашего подхода к управлению и стратегий тестирования внутренним процедурам Quogue мы с Барбарой начали проводить регулярные совещания с Натальей и координировать работу наших команд.

Самая запоминающаяся встреча прошла у нас в обеденное время прекрасным июльским днем – одним из тех редких для Нью-Йорка летних дней, когда влажность невысока, прохладно, воздух чист, а океанский бриз напоминает о том, что этот город – порт. Оставаться в здании было невыносимо, мы купили в столовой еду и отправились с ней на площадь перед Всемирным торговым центром.

Наталья была немного старше меня и Барбары, на вид ей было за сорок. У нее было двое детей, дочь только что поступила в университет, а сын учился в седьмом классе. Это была невысокая полноватая женщина, с широким лицом, глазами чуть навыкате и кудрявыми каштановыми волосами. Она была с Украины и говорила на английском свободно, но с сильным акцентом. Наталья жила с семьей в Бруклине и в тот момент занималась покупкой дома в Статен Айленде[1]. Разработанное ею автоматическое регрессионное тестирование было самым компактным и эффективным из всех подобных методов, с которыми я когда-либо сталкивалась.

Поскольку погода располагала, мы быстро расправились со служебными вопросами и перешли к обсуждению личных тем. Мы говорили о своих дочерях, о работе и личной жизни, о сбалансированности этих

[1] Статен Айленд (Staten Island) – один из пяти районов Нью-Йорка, расположенный на одноименном острове. – *Прим. перев.*

вещей, о трудностях с образованием. В конце концов мы спросили у Натальи, почему она уехала из Украины.

«Я из Чернобыля», – ответила она.

Нам потребовалось некоторое время, чтобы осознать этот факт.

«Вас эвакуировали?» – спросила Барбара.

«Через какое-то время. Но много дней мы оставались там. Правительство делало вид, что ничего не случилось. Мы узнали о том, что произошло, из радиопередачи по БиБиСи. Мы уехали, взяв с собой по одному чемодану, и больше уже никогда туда не вернулись».

«Наверное, было очень страшно, – сказала я. – И твое здоровье...»

«Пока все в порядке. У многих из моих бывших соседей был обнаружен рак. Я волнуюсь не столько за себя, сколько за своих детей».

Мы с горечью подумали о беспомощности матерей перед бедами этого мира.

Вдруг Наталья улыбнулась и указала в сторону сцены на западной стороне площади: «Смотрите! Сейчас будут выступать музыканты!»

В летние полдни Всемирный торговый центр часто давал возможность выступить группам: один день это был джаз, другой – рок, третий – старые хиты, четвертый – кантри. Поскольку к микрофону подошла высокая женщина в ковбойской шляпе и с электрогитарой через плечо, я поняла, что сегодня день музыки кантри. И, конечно же, вскоре она под аккомпанемент басиста, клавишника и ударника запела звонкую и бодрую песню. Перед сценой мужчина в тюрбане, женщина азиатской наружности, двое подростков-латиноамериканцев и пожилая белая парочка пустились в пляс. За ними последовали и другие – кто-то осторожно, кто-то с размахом – но все с радостью.

«Мне нравится музыка, – сказала Наталья. – Я такой прежде не слышала».

«Обычно я больше люблю джаз, – ответила я. – Но эта мелодия действительно зажигательная».

«Меня она несколько пугает», – сказала Барбара.

«Почему»? – спросила Наталья.

«Ну, мои предки родом из Северной Каролины, и когда мы слышали такую музыку, то обычно переходили на другую сторону дороги от греха подальше».

«Почему?» – еще раз спросила Наталья.

«Потому что... – музыка усилилась. – А, позже объясню».

Следующая песня была о женщине, которая решает плюнуть на все, пакует вещи в багажник и бесстрашно мчится по шоссе. Устоять было невозможно, и мы все втроем присоединились к танцующим.

После Дня труда[1], когда все уже начали входить в рабочий ритм после летних отпусков, сотрудники Quogue эффективно сотрудничали с нашей командой каждый на своем проекте, а работа по проектированию новых баз данных отсчитывала все новые контрольные сроки по графику, главный вице-президент компании «Фараон» по производству объявил о крупной внутренней перетасовке кадров, которая должна была коснуться всех, кроме ИТ-отдела. Поскольку компания расширялась, количество людей в офисах удваивалось, и в каждом свободном уголке и закутке организовывалось рабочее место. Руководитель административного отдела надеялся решить эту проблему, размещая всех в менее просторных, но более эргономичных помещениях. Сотрудники Quogue, которые были произвольно раскиданы по всему зданию, теперь должны были разместиться в бывших складских помещениях банка. ИТ-отдел не трогали, поскольку архитектор, посмотрев на наши требования по электропроводке и кабелям, сказал, что это слишком сложно, однако мы понимали, что хаос среди наших конечных пользователей внесет нестабильность и в наши проектные планы.

На последовавшем за перестановкой совещании у Дэйва все исполнительные директора в своих отчетах сообщили о задержках и проблемах в их проектах. Однако и две недели спустя ситуация не улучшилась. Собранные мною данные, а также мнения всех разработчиков, с которыми я общалась, указывали на то, что в большинстве случаев проблемы возникали по вине сотрудников Quogue.

«Были некоторые проблемы, – радостно согласился Фрэнк. – Но наше новое помещение гораздо тише. Там столы побольше и кресла поудобнее. Я со своего места вижу всех и могу отслеживать их работу. У нас сейчас идет период притирки и отладки, но, думаю, мы вскоре заработаем в полную силу».

К сожалению, это уверенное обещание оказалось слишком оптимистичным. Сроки срывались. Количество отказов и ошибок росло. Сотрудники Quogue, работоспособностью и дисциплиной которых прежде все восхищались, начали все чаще уходить на больничные. Проходя по коридорам, Фрэнк угрюмо смотрел на носки своих ботинок.

После запуска проекта разработки новых баз данных мы каждую среду собирались на совещание в самом большом конференц-зале отдела.

---

[1] День труда (Labor day) – национальный праздник в США, первый понедельник сентября. – *Прим. перев.*

Представители всех команд, работавших над задействованными системами, сообщали о своих успехах и проблемах. Разработчики баз данных обсуждали изменения, которые они вносили в следующую версию, планировавшуюся к установке на сервере обеспечения качества. На совещании присутствовали и тестировщики с менеджерами проектов – чтобы быть в курсе. Это было длинное, скучное, но необходимое мероприятие.

Через шесть недель после большого переезда на совещании в среду Наталья раздала документы, на которых были какие-то черные пятна. Так получилось, что после совещания все вышли, а мы с ней остались в комнате вдвоем, пока собирали вещи.

Я показала на черные пятна на ее документах.

«Может, вам нужно принтер починить?» – спросила я.

«Нет, это копоть».

«Какая еще копоть?»

Она зажала рот рукой с испуганным видом и, качая головой, растерянно привстала.

«Ничего, это ничего», – твердо сказала она.

«Нет, погоди. Что за копоть? Что случилось? Почему ты расстроилась?»

К моему удивлению, она упала обратно в кресло, закрыла лицо руками и начала всхлипывать.

Тут все и выяснилось. Старую систему вентиляции, установленную еще во времена банка, не прочистили перед тем, как завезти в помещение новую мебель. Когда ее включили – уже когда все переехали, – посыпалась грязь, накапливавшаяся в системах десятилетиями. Каждое утро работники Quogue вынуждены были как минимум полчаса счищать толстый слой копоти со своих столов, мониторов и кресел. К концу дня накапливался новый слой такой же толщины. Все бумаги нужно было хранить в ящиках.

«Все кашляют, – всхлипывала Наталья. – У Викрама сильные астматические приступы, а у Ольги начался бронхит».

«Как мог административный отдел допустить такое? – сказала я. – Это же нарушение Закона о труде».

«Какого закона?»

Я рассказала ей вкратце о Министерстве профессиональной безопасности и здравоохранения США, о Законе о труде и о защите прав работников. Она выслушала это все со скептическим видом.

«Все эти принципы очень хороши, но они касаются только граждан этой страны», – сообщила она мне.

«Но это не так. Почему ты так решила? Мы все исправим. Я поговорю с Дэйвом, и мы назначим встречу с менеджером административного отдела».

«Нет, пожалуйста, не нужно! – простонала она. – Я потеряю работу. Мне нужно будет вернуть 50 тысяч долларов, а мы только что купили дом...»

«Наталья, о чем ты говоришь?» – от этих страшных пророчеств у меня начала закрадываться мысль, что, возможно, после трагедии на атомной станции у нее развилась паранойя.

Но нет: оказалось, у нее были серьезные причины бояться. Она объяснила, что трудовой договор в Quogue заключался на базе двухлетнего «испытательного срока». Если сотрудника увольняли в этот период, он должен был возвратить компании полную сумму, выплаченную ему с момента приема. Иммигранты рисковали также потерять работодателя, который выступал спонсором их гринкарты, и могли быть депортированы. Фрэнк запретил своим сотрудникам жаловаться на их новые рабочие места, четко дав понять, что нарушитель будет уволен.

Закончив, Наталья устало откинулась на спинку кресла. Она уже не плакала, но смотрела на меня обреченно. Никогда прежде не лежал на мох плечах такой эмоциональный груз – взрослый человек вверял свою жизнь в мои руки.

«Хорошо, – сказала я. – Я поняла. Этого разговора не было. Но я решу эту проблему».

Я вернулась в свой кабинет, закрыла дверь и пошла искать музыку, которая помогла бы мне думать. В пачке дисков я нашла недавно купленный альбом той группы, которую мы слушали перед Торговым центром. Под аккомпанемент воодушевляющих песен я позвонила Барбаре.

«Ты когда-нибудь заходила в банковские помещения после того, как туда въехали Quogue?» – спросила я.

«Нет, они всегда сами приходят на встречи. И всегда так было, даже до переезда. А что?»

«Ну, мне подумалось, что, может быть, они никак не освоятся на новом месте, потому что у них не было вечеринки по поводу новоселья?»

Она засмеялась: «Не думала, что ты столь суеверна».

«Я подумала, что будет хорошим дружеским жестом принести им вкусностей к завтраку завтра утром. Присоединишься?»

«Конечно. Что принести?»

В 8.00 мы пришли к запертой двери с пончиками, багетами, круассанами, соком и фотоаппаратом. Я набрала код на двери, но он больше не срабатывал.

«Странно, – сказала Барбара. – Ты получала письмо о смене кода?»

В этот момент подошел Рафик и впустил нас. Сначала он был просто рад нас видеть, а потом посмотрел на нас с благодарностью.

Мы прошли по небольшому коридору к открытому пространству у копировальной техники и поставили свои коробки на шкафчик.

«Доброе утро, Квогги! – крикнула я. – Завтра подан!»

Люди выглянули из своих отсеков.

«О Боже, – выдохнула Барбара. – Ну и грязища здесь!»

Слой копоти покрывал столы и шкафы ровным слоем, можно было пальцем написать свое имя. Рядом с каждой группой отсеков стояли пачки бумажных полотенец. Многие сотрудники были в черных свитерах или шалях, защищающих одежду от грязи, на головах у них были шарфы и шапки. Мелкий черный снег летел из вентиляционных отверстий над головами.

«Похоже, с вентиляцией проблемы», – отметила я.

Барбара стряхнула пыль со своей желтой шелковой блузки: «Где наш фотоаппарат?»

Фрэнк примчался вприпрыжку из своего застекленного кабинета и пытался заслонить объектив.

«Леди! Как любезно с вашей стороны. Спасибо за угощение».

Барбара обошла его и начала фотографировать. Я обратила внимание, что вентиляционное отверстие в его кабинете было затянуто самодельным марлевым фильтром.

«А что говорит административный отдел по поводу ситуации с очисткой воздуха?» – спросила я его.

«Они работают над этим. Ищут особого подрядчика, который сможет почистить воздуховоды»

«А с Дэйвом вы на эту тему говорили?»

«Да, он сказал мне следовать указаниям административного отдела, – пожал он плечами. – Бывало и похуже».

«Могу представить».

Фотографии не могли передать всего, и мы с Барбарой взялись сами рассказать обо всем этом. Мы поговорили со всеми менеджерами по разработке и ведущими разработчиками, в команде которых работали сотрудники подрядчика, и большинство из них нашли время сходить на рабочие места Quogue. Я внесла этот пункт в повестку следующего совещания у Дэйва. Я была уверена в том, что он признает срочность этой проблемы, и была готова обсудить несколько альтернативных предложений.

Первые результаты были мягко говоря неутешительными.

«Дэйв, я уверена, что есть прямая связь между проблемами с качеством воздуха на новых рабочих местах и спадом ключевых показателей продуктивности, – сказала я, когда мне дали слово. – Я хотела бы порекомендовать нам всем...»

«Что ты думаешь?» – спросил Дэйв у Фрэнка.

«Не вижу связи, – сказал Фрэнк. – Хотя будет очень хорошо, когда воздуховоды почистят».

«Это вопрос к административному отделу, – сказал мне Дэйв. – И Фрэнк этим занимается».

«Я понимаю это, и знаю, что он старается решить этот вопрос, но мне кажется...»

Дэйв поднял палец, чтобы остановить меня.

«На будущее, – сказал он сурово, – я хотел бы, чтобы вы запомнили, что ваша работа по обеспечению качества касается только программного комплекса».

Я оглянулась в поисках поддержки и увидела стену из непроницаемых лиц. Мне живо представился звук удара – это моя карьера с размаху налетала на скалу.

После совещания я мрачно шла к дамской комнате, когда Марк поймал меня за руку.

«Погоди, – сказал он. – Это был только первый раунд».

За его спиной согласно кивал головой Зви.

Второй раунд представлял собой вооруженную публичную кампанию. По следам Барбары, менеджеры проектов по всем приложениям, у которых работали привлеченные сотрудники Quogue, организовывали встречи с людьми, от которых зависело финансирование проектов, описывая отрицательное влияние проблемы с помещением на процесс разработки. Ник, руководивший, в том числе, правовыми системами,

получил задание проверить договоры, заключаемые в Quogue, и выяснить, применимы ли правила, установленные Министерством профессиональной безопасности и здравоохранения США к сотрудникам подрядчика, работающим в помещениях «Фараона». Кэрол позвала вице-президента по информационным технологиям отметить день рожденья в своей группе, а затем заманила его в помещение, где работали сотрудники Quogue. Под предлогом покупки стола для пинг-понга в фитнес-центр Марк зашел к руководителю административного отдела и спросил у него о планах по очистке воздуховодов.

Дэйв не обрадовался, увидев вопрос о качестве воздуха вновь на повестке дня уже на следующем совещании. В этот раз его добавил Марк, меня убрали с линии огня.

«Не знаю, почему мы все еще обсуждаем это, – сказал Фрэнк. – Многие вопросы адаптации к новым рабочим местам уже решены, показатели работы улучшились».

Отчасти он был прав. Почувствовав облегчение от того, что кто-то обратил внимание на их беды, люди из Quogue выполнили некоторые задачи по проектам «Фараона», взятые ими еще до переезда. К сожалению, это им не помогло.

«Мы все еще обсуждаем это, потому что это позор», – отрезал Марк.

Дэйв сказал Фрэнку, что тот свободен. Когда он вышел, Дэйв свирепо повернулся к Марку.

«Моральный контекст, который ты пытаешься притянуть в этой ситуации, выходит за всякие рамки, – сказал он. – Административный отдел дал нам понять, что если мы хотим, чтобы вентиляционную систему почистили, нам придется заплатить за это, а в наш бюджет это не заложено».

«Мы не должны никого заставлять работать в таких нездоровых условиях», – сказал Марк.

«А кто их заставляет? – спросил Пол. – Это надежная хорошая работа. Если она им не нравится, они всегда могут уйти».

«Они не могут уйти, – сказала я. – У них нет выбора. Они привязаны своими договорами».

«У всех есть выбор, – проворчал Пол. – Каждый делает его, а потом принимает на себя его последствия».

«На какой период у нас заключен договор с Quogue? – спросил Анупам. – Если мы отдадим услуги на аутсорсинг в Индию, нам не придется беспокоиться об американских законах и стандартах».

Зви развел руками: «Это ошибка. Архитектор, руководитель административного... они забыли, мы забыли... теперь уже не найти виновного; дело сделано. Нужно все исправить. Давайте соберем все стороны вместе и найдем выход из ситуации, а потому уже решим, кто будет платить».

«Зви прав, – Кэрол Чу не улыбалась. – Но деньги не должны идти из бюджета разработки систем. Наш босс по информационным технологиям будет против».

«Поверь мне, наш главный не собирается протестовать, – вздохнул Дэйв. – Послушайте, мы имеем дело с временными работниками по найму. Их наняли исключительно для того, чтобы сократить расходы. Мы же не на мясокомбинате».

Я как новичок в политических боях предположила, что наш крестовый поход за чистоту среды окончился шахом и матом. Но Кэрол Чу заверила меня, что игра еще не закончена. Закопченные воздуховоды помещения Quogue стали главной темой слухов в «Фараоне». Волна общественной симпатии и сочувствия к нашим подрядчикам уже начала вызывать некоторый резонанс и в высших кругах.

В третьем раунде участвовали уже руководители высшего звена: главный вице-президент по производству, главный вице-президент по информационным технологиям и главный юрисконсульт; и происходило оно на выездном заседании по стратегическому планированию. Сподвигло ли их на это чувство справедливости или страх того, что начнутся трудовые споры и все это просочится в средства массовой информации, но они решили, что проблема достаточно серьезна, чтобы немедленно ею заняться. Как только они вернулись в офис, главный вице-президент по производству сменил руководителя административного отдела и нанял подрядчика, специализирующегося на системах вентиляции. Главный юрисконсульт начал переговоры с архитектором здания о том, кто должен оплачивать счета за очистку воздуховодов. Главный вице-президент по информационным технологиям дал указание написать статью в еженедельный внутрикорпоративный выпуск новостей, восхваляющую Дэйва за его гуманистические принципы и его способность соединять финансовую ответственность с заботой о здоровье и благополучии всех его работников.

Руководство Quogue Consulting выразило свое неудовольствие действиями Фрэнка в этом эпизоде, отозвав его на несколько дней в свою штабквартиру для расследования и воспитательных мер.

Во время отсутствия Фрэнка я получила сообщение от Рафика, в котором он просил, чтобы я заглянула к ним и посмотрела результаты тестирования на его компьютере.

Теперь бывшее банковское помещение было в безупречном порядке. Две недели по выходным подрядчик выгребал грязь, накопленную за десятилетия. Больше из вентиляционных отверстий не летела копоть. Утреннее солнце весело блестело на всех поверхностях, которые были так же чисты, как и предыдущим вечером. Сотрудники Quogue сбросили свои защитные черные одежды.

Выяснилось, что результаты тестов были только приманкой, а меня ожидал сюрприз – небольшое торжество. В отсутствие начальника разработчики и тестировщики собрались вокруг копировальной машины. Рафик подвел меня к ним. Наталья скромно протянула мне конверт с открыткой. На открытке была надпись «Спасибо». Внутри каждый из них написал свое имя. Позже, показав открытку моей юной и не впечатленной этой историей дочери, я попыталась объяснить ей, что не все самые замечательные деловые достижения можно изложить в резюме.

Итак, мы вылечили больную часть здания. Жизнь вернулась в привычное, нормальное русло, как для работников «Фараона», так и для наших подрядчиков из Quogue. Мы завершили проект по проектированию новых баз данных – не уложившись ни в сроки, ни в бюджет, но результат удовлетворял производственной необходимости, и мы чувствовали, что заложили прочный технологический фундамент на будущее.

Однако будущее нашего здания и наших людей вовсе не было ни прочным, ни спокойным. 11 сентября 2001 года террористы атаковали здания Всемирного торгового центра, на крышу нашего офиса посыпались обломки самолетов и человеческие тела, и все вокруг окуталось облаком едкого дыма.

Всю следующую неделю вокруг царил хаос, и никто не работал. Через неделю мы вернулись к работе в резервном помещении для работников ИТ на другом берегу Гудзона в Джерси.

Из окна конференц-зала был виден горящий щебень и угол нашего здания. Нас разместили в просторном помещении, похожем на биржевую площадку, только мониторов на каждое рабочее место поменьше. Никто из наших сотрудников не погиб в теракте, но у каждого были знакомые, которые кого-то потеряли. Наш новый руководитель административного отдела, здоровый 43-летний мужчина внезапно умер от сердечного приступа 15 сентября при осмотре ущерба, нанесенного нашему зданию.

Нам всем тогда было немного не по себе.

На первом совещании у Дэйва мы смотрели, как из котлована поднимается жуткий дым, обсуждая коды отсутствия в базе данных отслеживания проекта ОУП.

«Будем кодировать последнюю неделю как дни на больничном?» – спросил Фрэнк.

«Думаю, больше подходит „пропуск по личным причинам“», – ответила Кэрол Чу.

«Но тогда эти дни будут вычтены из резерва каждого сотрудника», – заметила Барбара.

«Это отразится на кадровой системе и системе учета рабочего времени», – сказала я.

«А нет ли категории отсутствия по независящим от работника обстоятельствам, снежные заносы, например?» – предложил Зви.

«А что по поводу сверхурочных?» – спросил Пол.

«Если говорить о сверхурочных, – сказал Марк, – я хотел бы быть уверенным, что Наталья получит особое вознаграждение за то, что она делает».

«Мы *все* делаем больше, чем в человеческих силах», – огрызнулся Дэйв.

«Это правда, – заметил Зви. – Но Наталья, по сути, содержит гостиницу. У нее на полу спят восемь разработчиков – наших и из Quogue».

«Почему?» – просил Фрэнк.

«Они живут в Бруклине, весь дым идет в ту сторону, – объяснил Марк. – А ее дом находится в Статен Айленде. Это с наветренной стороны от дыма».

«Мы никогда не отмечали отработку сверхурочного времени, – сказал Дэйв. – И не будем начинать это сейчас».

Наш спор был прерван рычанием и громкими звуками ударов. Рефлекторно вся команда руководства вскочила из-за стола, чтобы выяснить, что случилось. Мы обнаружили, что в холле какой-то мужчина атаковал торговый автомат.

Это был Доминик, разработчик на Java, работавший с системами счетов к оплате. Средних лет, бритый наголо, круглолицый, с крепкой фигурой борца, он в ярости обрушил на автомат складной металлический стул, а затем еще несколько раз хорошенько врезал по нему ногой.

«Отдавай мои чертовы деньги! Хочу шоколадку!!» – ревел он.

Марк печально покачал головой:

«Бедный Доминик. У него дядя пожарный. Он сказал мне, что за прошлую неделю побывал на семи похоронах».

Другие люди, очутившиеся в тот момент в холле или поблизости от него, расположились полукругом на безопасном расстоянии.

Доминик повернулся и посмотрел на них.

«Я проел последний доллар!»

Зеваки отступили на шаг, опасаясь, что он швырнет в них стул.

«Я вызову охрану», – сказала Барбара.

«Не нужно, – сказал Пол. – У него и так хватает проблем».

«Посмотрим, что я смогу сделать», – сказал Марк.

Но не успел Марк двинуться, как от группы людей отделилась Наталья и спокойно пошла прямо к Доминику.

«КАК Я ЗОЛ!» – заорал он так, что я вздрогнула.

Но не она. Она протянула руку и погладила его по щеке.

«Да, – сказала она. – Конечно, ты зол. Я понимаю».

И как будто лопнул нарыв. Гнев Доминика тут же стал улетучиваться. Он поставил стул. Через мгновенье он с испугом оглядывал вызванный им переполох.

«Пойдем», – сказала она.

Отвела его на кушетку в холле, села рядом и приобняла за плечи. Он сидел, упершись локтями в колени и спрятав лицо в ладонях, как олицетворение отчаянья.

В тот вечер мы с Натальей встретились у лифта, когда уходили домой. Пока ждали, мы обсудили свои наблюдения о том, как справляются со всем произошедшим наши дети. Мы согласились, что подростки не склонны рассказывать родителям о своих чувствах и за ними нужно внимательно присматривать.

«Знаешь, хотела тебя кое о чем спросить, – сказала я. – Можно ли сравнить эту катастрофу с тем, что было в Чернобыле?»

«О, почти ничего общего, – она печально улыбнулась и покачала головой. – В Чернобыле правительство делало вид, что все в порядке и в новостях ничего не сообщали. Мы понятия не имели, что происходит. Здесь сообщают все новости. Я очень рада жить в стране, где правительство не врет своему народу».

Мы вернулись в наше здание в конце октября. «Фараон» был одной из первых крупных компаний, возобновивших свою деятельность внутри охраняемой военными зоны. Того же подрядчика, который чистил системы вентиляции в помещении Quogue, теперь наняли для куда большей работы по очистке здания, обеззараживанию и фильтрации воздуха. Несмотря на едкий запах горелой плоти и других неведомых веществ, который накрывал волной, стоило только выйти из здания,

воздух в округе был признан Агентством охраны окружающей среды безопасным.

Жизнь неотвратимо шла вперед. Через год я ушла из «Фараона», став консультантом и преподавателем. Анупам все-таки смог убедить высшее руководство в том, что зарубежный аутсорсинг – хорошая идея. Когда закончился контракт с Quogue Consulting, их проекты были переданы индийской компании и выполнялись в Бангалоре. Доля участия той компании стремительно росла, и сотрудники «Фараона» попрежнему часто увольнялись.

Большинство моих бывших коллег продолжили работать в сфере ИТ в издательской или финансовой отрасли. Скотт уехал в Австралию и открыл бизнес по организации дайвинговых туров на Большой барьерный риф. После развода Дэйв занял весьма высокопоставленную должность в ведущей консалтинговой компании, выдавая клиентам рекомендации о том, как сократить расходы за счет передачи работы зарубежным подрядчикам. Барбара стала директором отдела управления проектами в компании, разрабатывающей и производящей несколько брендов одежды высшего класса.

Здания нашего офиса больше нет. Пережив теракты, оно пало жертвой рынка недвижимости Манхэттена. Его снесли, и на его месте сейчас красуются шикарные многоквартирные дома.

Единственный человек, о потере связи с которым я жалею, это Наталья. Я часто о ней думаю, особенно когда в программе курсов в бизнес-школе, где я преподаю, есть изучение командной работы. Надеюсь, однажды мы найдем друг друга в какой-нибудь социальной сети. Я хочу знать, как поживает она и ее семья. И еще я хотела бы снова задать ей тот же вопрос, что задала тогда у лифта, и узнать, верит ли она попрежнему в эту страну...

30

Обобщая мнение команды

Энди Орам

Компании возникают и исчезают. А ценность, которая никуда не уходит, – это люди, работающие в них. Люди приходят на работу со своим личным жизненным опытом, а не только с техническими умениями, они привносят в работу межличностные отношения, которые выстраиваются между сотрудниками и руководителями за долгие годы.

В этой связи мы можем на одну чашу весов поставить важность формальных, заявленных официально составных частей проекта, а на другую – кажущиеся периферийными впечатления, которые команда получает в процессе работы. Я уверен, что вы вспомните очень мало проектов за свою карьеру, результатами которых все еще пользуются. Учитывая это наблюдение, я всегда чувствовал, что влияние какого-либо проекта на развитие рабочей команды столь же важно, как и сформулированная, поставленная при разработке проекта задача. В действительности, я считаю, что именно это влияние лежит в основе данной книги.

Наибольший потенциал этот принцип несет в себе в тех исключительных случаях, когда проекты обречены на провал. Я имел несчастье работать как раз над таким проектом лет двадцать назад. Единственным положительным результатом этой работы, который я могу теперь вспомнить, был момент просветления, когда я собрал воедино мудрость всей команды и выразил ее, дав ей прозвучать.

В таких случаях личное выживание становится триумфом, поскольку подобные проекты бросают вызов вашему умению профессионально мыслить ежеминутно: это и необходимость приспосабливать плохо продуманные решения к долгосрочным обязанностям команды, поддер-

живать своих друзей и свою цельность в условиях невыносимого давления в планировании, слушать и соглашаться с предложениями, про которые все знают, что они невыполнимы, а потом вернуться на свое рабочее место и подогнать их к реальным условиям работы.

Кроме того, проекты, которые угрожают выживанию, подтачивают хрупкие границы «я», которое отвечает за культуру поведения, провоцируя растворение в неистовом безумии. Я однажды дошел до этого предела, работая над проектом, который описываю в этой главе. Но смелое предложение, внесенное одним из моих менеджеров, удивительным образом направило меня от агонии к восстановлению равновесия. Полагаю, тот факт, что я дал возможность высказаться конкретному участнику проекта, в некотором смысле помог всем нам выжить.

Теперь, глядя назад и размышляя над этой храброй выходкой, я вижу в ней также и зерно нового типа взаимоотношений между отдельными участниками проекта и руководством. Мы с моим менеджером наощупь продвигались к пониманию корпоративного поведения, которое в данный момент активно обсуждается ведущими консультантами в сфере менеджмента на страницах деловых журналов и книг, занимающих полки книжных магазинов. Мы мечтали о командной структуре, которая не имела бы четкой вертикали, а больше распространялась вширь, основываясь на прямом общении между членами команды, стоящими на разных ступенях иерархической лестницы, и исследовали возможности цифровых технологий, чтобы реализовать такую структуру на практике.

Короче, этим импульсивным жестом я предвосхитил то видение командной работы, которое два десятилетия спустя было сформулировано экспертами организаций.

Сокровища из компьютерного прошлого

Закройте глаза и при помощи памяти, если вы достаточно долго работали в компьютерной индустрии, или при помощи воображения, если у вас такого опыта нет, вернитесь в прошлое, в более простое и невинное, когда небольшая компьютерная компания могла создать с нуля полноценную систему и продать ее на рынке. В 1970–1980-е, имея в распоряжении компьютерные чипы и другие компоненты в качестве «строительного материала» и рабочую команду из ста человек, компания, которую я назову Edom Engineering, создала систему UNIX, обладавшую необычайно мощными возможностями по обработке данных. Ее можно было рассматривать как поставщика недорогих систем, удовлетворяющих самым современным технологиям, и она пользовалась

популярностью в небольших научных лабораториях и конструкторских бюро, планы которых сильно опережали бюджет.

Неординарные личности, работающие в Edom Engineering, многих из которых вы встретите в этой истории, превращали работу в стимулирующий и приносящий удовлетворение процесс. Я дорожу воспоминаниями о сотрудничестве с этими людьми, с которыми пережил столько критических ситуаций. Очень возможно, что многие из них прочитают эту историю и узнают ситуацию, которую я здесь опишу. Надеюсь, они простят меня в том случае, если их воспоминания об этих событиях расходятся с моими. Я сделал все возможное, чтобы передать чувства и факты того времени, хотя прошло уже два десятка лет.

Компания Edom Engineering располагалась в подсобном помещении с низкими потолками на Втором технологическом шоссе Бостона (Шоссе 495, в отличие от более известного Шоссе 128). На нижнем этаже располагались производственные помещения, на верхнем – команда проектировщиков. Компания принадлежала к высшему обществу, но вместе с тем была гармоничной и эффективной. Количество и разнообразие аппаратного обеспечения, от материнских плат до оборудования для сбора данных, создавалось практически из ничего и было специально спроектировано для специализированного системного ПО, компиляторов и других программ.

Сегодня совсем немногим рабочим командам приходится вести диалог, участником которого я был в то время, когда программисты, создающие программы-компиляторы, обсуждают баги в эвристиках назначения регистров и вопрос о том, как добиться еще большей точности при вычислении квадратного корня. Инженеры компании Edom Engineering, многие из которых были докторами наук, представляли умную команду с отличным командным духом.

Перед нашей группой программистов, разрабатывающих компилятор, была поставлена задача воспроизвести все черты и свойства классического доминирующего на рынке компилятора FORTRAN, разработанного компанией Digital Equipment для их же мини-компьютеров VAX. Одна из участниц группы разработчиков настолько увлеклась заданием, что вывела алгоритм, применяемый в генераторе случайных чисел компании Digital Equipment (хотя фактически это был генератор псевдослучайных чисел), сделав так, что наша функция `RAND` генерировала такую же последовательность чисел, как и функция `RAND` библиотеки Digital.

Помню, как спросил у нее: «Неужели требования, выдвигаемые проектом, заставляют тебя добиться той же последовательности случайных чисел?»

На что она, пожав плечами, ответила: «Я просто разгадала этот алгоритм и решила, что могу сделать так, чтобы наша библиотека тоже соответствовала нормам, насколько это возможно».

Просто это была такая смесь яркого организаторского таланта, стремления быть лучшими и совершенной радости от творчества, которая в результате привела команду Edom Engineering к полному переоснащению ОС UNIX и других программных компонентов для максимального повышения эффективности хорошо зарекомендовавшего себя аппаратного обеспечения компании. Наша команда маркетинга гордилась сильнейшей производственной поддержкой широкого спектра функций обеих систем – System V и BSD UNIX, которые на тот момент были двумя фундаментальными источниками всех систем UNIX.

Проекты оформлялись вокруг популярной в 1970-е междисциплинарной модели, на которую сильно повлияло впервые примененное компанией Digital матричное управление. Разработчики из каждой команды взаимодействовали лично с сотрудниками отделов маркетинга и техподдержки и с разработчиком документации, совместно работавшими над проектом. Эти участники посещали все встречи по проекту и следили за работой друг друга почти ежечасно.

Забота компании заключалась в создании хороших технических руководств, а уважение к команде по составлению документации навело меня на мысль присоединиться к ним в качестве разработчика документации. Меня чаще всего ассоциировали с командой программистов, пишущих компилятор, но мне также было приятно участвовать и в других проектах.

Несомненно, технические руководства компании Edom Engineering были известны везде в отрасли производства в силу все того же качества, которое характеризовало другие продукты компании. Программисты на самом деле понимали наши технические руководства к драйверам и, прочтя их, могли работать эффективнее. Первый год я посвятил созданию документации по всему языку FORTRAN, которое может показаться пустым упражнением в создании избыточной информации, но это на самом деле было необходимо, поскольку мы обогатили стандартный FORTRAN множеством расширений VAX, усовершенствовав его по ряду параметров.

Стиль руководства в команде для того времени отличался свободой и демократичностью. Мы придерживались свойственной стартапам неряшливой схемы управления даже после того, как численность персонала перевалила за 300. Но такой упор на открытость таил в себе серьезную проблему: топ-менеджеры были намного теснее связаны с инвесторами, чем с рабочей командой. Недавнее слияние еще больше увеличило дистанцию, отделявшую инвесторов от рядовых членов команды. Эта

дистанция порождала постоянные жалобы и грустные шутки среди подчиненного персонала компании. Но когда надо было выслушать своих подчиненных в связи с важными проектными решениями, менеджерам и инвесторам всегда хватало сообразительности дать слово техническим разработчикам, не имеющим равных в понимании вопроса.

В период расцвета компании Edom Engineering казалось, что разработчики могут все. Когда парадигмы многопоточности стали важным элементом программирования, команда разработчиков компилятора добавила в компилятор C поддержку стандартных парадигм многопоточности всего за пару месяцев, а графическая группа успешно разработала возможность безопасного многопоточного запроса в библиотеке Motif (ведущая графическая библиотека UNIX в то время), когда проект стал жертвой производственного кризиса, в который компания попала в конце 1980-х.

Как раз об этом экономическом кризисе и о саморазрушительных действиях менеджеров компании я и хочу здесь рассказать. Наша индустрия прошла через серьезные, катастрофические изменения, которые отнюдь не являются редкостью. Такие изменения в парадигме программирования моим читателям придется неоднократно пережить за свою карьеру – неважно, в какой сфере они работают. И эти изменения заслуживают того, чтобы посвятить им несколько слов и помочь вам понять, насколько ограниченны были варианты действий в нашем случае.

Изменения

В 1980-е росли требования и ужесточалась конкуренция в нашей отрасли разработок, которую можно назвать научной обработкой данных. Поскольку аппаратное обеспечение становилось быстрее, обычные рабочие станции стали лучше удовлетворять требованиям потребителей, при этом не нужно было прибегать к ухищрениям, которые мы использовали в Edom Engineering, чтобы добиться лучших результатов. Параллельно с этим стандартные и расширенные функции ОС UNIX дошли до того, что даже наша сумасшедшая команда программистов не могла реализовать все то, чего он нас ждали потребители.

Наш основной рынок сбыта научной и инженерной продукции тоже стал сужаться, и причиной был совершенно не связанный с ним фактор: в 1980-е сократилось финансирование американской армии, за счет которого существовали многие наши потребители.

В конце 80-х менеджеры компании Edom Engineering решились на поступок, который, как они надеялись, положит начало новой базе усовершенствованных компьютеров. Наша компания всегда строила свою систему на чипах Motorola. Но новая волна процессоров для RISC-машин

(вычислительных машин с сокращенным набором команд) предлагала намного более высокую производительность. Мудрецы из нашей команды разработчиков аппаратного обеспечения изучили возможности этих процессоров и выбрали один из них, который, по их мнению, мог дать ту производительность, которая была нам нужна.

Причина популярности RISC-процессоров заключалась в возрастающей многоуровневости принятия решений, чего не хватало на тот момент чипам Intel и Motorola, которые компьютерщики сегодня называют CISC (Complex Instruction Set Computing – процессоры со сложным набором команд). Традиционные чипы получили это название, поскольку обладают множеством инструкций, часть которых рассчитана на выполнение очень узкого круга задач. Разные инструкции требуют разного количества циклов, поэтому разработчикам аппаратного обеспечения, с одной стороны, и создателям ПО с другой, приходится разрабатывать все более сложную планировку задач. Влияние всех этих компонентов привело к тому, что требовалось все больше проводов, соединяющих процессор с памятью.

К 1980-м компьютерщики определили, что возрастающее количество служебных протоколов и сигналов снижает реальную мощность процессора, придя к выводу, что нужно придумать новое поколение конкурентоспособных процессоров, используя сведенный к минимуму набор простых коротких инструкций, сходных между собой по количеству необходимых циклов для их выполнения. Десять лет на рынок поступали разработки разных компаний. RISC стал основной темой, которая обсуждалась в прессе, и его критерии выдержали предсказания разработчиков.

Так было заложено начало проекта Longjump в компании Edom Engineering. В этом проекте с самого начала было много слабых мест. Мы разрабатывали систему на базе нового продукта, основанного на новой парадигме расчетов, и зависели от маленькой фирмы, история которой нам была неизвестна. В то же время, пока команда разработчиков аппаратного обеспечения пыталась написать под новый процессор новую, яркую и оригинальную систему, используя все преимущества всех функций для повышения скорости работы, команда программистов ПО должна была создать уникальную, хорошо отлаженную операционную систему, компилятор и библиотеки. Поскольку RISC-процессор сильно отличался от привычных чипов Motorola, нам пришлось создавать свои системные утилиты на основе тех, которые поставляли нам распространители RISC-машин. И эти усилия сильно всколыхнули полные неожиданностей воды.

Руководителем проекта Longjump был назначен один из разработчиков, которого я назову Джордж. Джордж был относительно молодым

руководителем проекта, он был добросовестным и неутомимым человеком, неординарной личностью, что в то время было редкостью для программиста. Он носил небольшие усы, немного сутулился, сидя на собраниях, и смотрел на окружающий мир, плотно сжав губы, которые сжимались еще сильнее, когда он слышал нежеланные новости. Он разбирался в маркетинге так же хорошо, как и в проектировании, и стремился ходить в строгих костюмах, в то время как типичный программист-компьютерщик его возраста одевался еще более невнятно и неряшливо, чем сегодня. Он, должно быть, знал, что более представительный вид повысит возможность благоприятного исхода его встречи с ведущим менеджером. А поскольку это было его первое назначение руководителем проекта, он хотел произвести хорошее впечатление.

На собрании компании, где представляли проект Longjump, Джордж встал перед собравшимся руководством и командой и произнес краткую речь, в которую вставил довольно типичную фразу: «Я бы хотел поблагодарить руководство компании за то, что они доверили мне управлять этим проектом».

Не думаю, что он предполагал реакцию команды на это заявление. Мы все знали, что Longjump – рискованный проект, и в этой ситуации нельзя было оставлять место даже для капли застенчивости. Поблагодарив руководство за доверие, он заронил в нас сомнение, сможет ли он оправдать себя в этой роли.

Но, в конечном счете, если Джордж и сыграл отрицательную роль в этой истории, то заключалась она в том, что он выказывал слишком много почтения своим начальникам, пока они вели проект в состояние застоя.

Преодоление

Моя философия, как я уже сказал в начале главы, ориентирована с одной стороны на общество, а с другой – на рабочий процесс. На общество она ориентирована, поскольку я верю в то, что любой успех или неудачу можно использовать на благо укрепления команды, а ориентирование на рабочий процесс важно, поскольку я верю в то, что любое решение по проекту должно учитывать то, как оно отразится на членах команды и на ресурсах.

Раньше, в более светлые времена, наша техническая группа разработчиков однажды проводила собрание (оно было похоже на одно из многих собраний, которые проводят программисты UNIX) по поводу реорганизации оперативных страниц руководства, представляющих собой, грубо говоря, собрание исторической документации UNIX. Через 45 минут после того, как мы успели рассмотреть вопросы о конку-

ренции и установить некоторые правила поведения, я сказал: «Важно не решение, к которому мы рано или поздно придем, а тот факт, что мы это здесь вместе обсуждаем».

Некоторые члены рабочей группы были шокированы расточительным подходом к вопросу о распределении нашего времени, но, думаю, это польстило менеджеру по написанию технической документации Алану. Он регулярно говорил, что хочет быть целиком и полностью собой на рабочем месте, включая эмоциональные реакции и личные качества. Однажды он попросил свою команду прочитать недавно изданную книгу о менеджменте, ключевая мысль которой заключалась в том, что компании должны основываться на дальновидности, чтобы иметь возможность насладиться длительным успехом в будущем. И эта дальновидность должна приносить ценный результат для потребителей, работников и для всего общества в целом.

Алан назначил меня руководителем группы по составлению документации к проекту Longjump – это была моя первая работа, как у Джорджа. Я решил, что мой долг – защищать команду насколько это возможно. У нас не было переговорной, чтобы обсудить рабочий процесс или результаты. Но я попытался защитить нас от утомительной бюрократии. Более того, я договорился о том, чтобы делиться с разработчиками документации всей информацией, которая может помочь предсказать наступление кризиса, прежде чем он успеет разразиться.

Так случилось, что у меня было, *что* предложить по последнему пункту. Как руководитель группы разработчиков документации я присутствовал на собраниях группы разработки ПО. Я вел подробный конспект, выделяя все, что могло быть ценным для моей группы авторов. Я сделал так, чтобы все авторы моей группы видели дорогу, по которой идут, далеко вперед.

У управления командой есть сторона, при мысли о которой стонет большинство разработчиков, – регулярные собрания. Обычно на таких собраниях члены команды скучно рассказывают о том, что успели сделать за неделю, а потом возвращаются на свои рабочие места, не обогащенные никакой новой информацией. Команде разработчиков Longjump я объявил, что каждую неделю буду оставлять повестку дня пустой, и мы будем собираться, только если у кого-то есть что-то, чем он хочет поделиться.

Это маленькое нововведение показало моим коллегам, что я стремлюсь к эффективности и демократии. Авторы-разработчики поблагодарили меня за возможность сократить эти бессмысленные встречи. Однако у каждого в моей команде было право назначить собрание, если было чем заполнить повестку дня.

На практике, я был единственным, кто заполнял пустую повестку дня. Когда мы все-таки собирались, я сообщал им новости, интересные для всей группы, и через 15 минут мы обычно расходились. Не прибегая к модным современным техникам проведения таких собраний, мы применили свою технику по максимально эффективному использованию своего времени.

Кодирование

Всегда полезно иметь в распоряжении несколько инструментов по поддержке изменений поведения в организации. Скромный пример: я поместил файл с пустой повесткой дня в общий каталог обмена файлами. Вероятность того, что два человека будут одновременно редактировать один и тот же файл, – почти равна нулю, но я настолько привык воспринимать все в формате операционных систем и многопоточных задач, что почуял здесь возможность возникновения конкуренции. В результате я с помощью простого сценария и команд вроде `chmod` блокировал доступ к файлу повестки дня, если его уже кто-то использовал.

Более глобальные сложности были с оперативными страницами руководства, которые я уже упоминал. Все наши языковые инструменты опирались на новый компилятор, предлагаемый нашим поставщиком. Мы решили, что адаптация наших оперативных страниц руководства к новой системе потребует слишком много усилий, и потому откуда-то сверху пришло распоряжение конвертировать для нашей системы оперативные страницы руководства поставщика.

Проблема заключалась в том, что поставщик предоставил нам эти страницы только в формате вывода, а для того, чтобы адаптировать их под нужды наших потребителей, нам необходим был исходный формат. Типичный формат вывода имел примерно следующий вид:

```
O^HO^HOP^HP^HPT^HT^HTI^HI^HIO^HO^HON^HN^HNS^HS^HS


-^H-^H-o^Ho^Ho _^Hf_^Hi_^Hl_^He
Output binary image to _^Hf_^Hi_^Hl_^He.
```

Это краткое описание классической опции `-o` компилятора. То, что в примере имеет вид `^H`, на самом деле является просто *символом возврата на одну позицию* (ASCII-символ 8), что заставляет терминал накладывать следующий символ на предыдущий. Тройное наложение символа в представленном здесь формате приводит к тому, что символ на терминале получается жирным. При наложении на символ подчеркивания символ выводится подчеркнутым.

Нам же был нужен исходный макрос troff, используемый для создания оперативных страниц руководства, в котором предыдущий текст выглядел бы вот так:

```
.SS Options
.TP
/fB-o/fP /fI-file/fP
Output binary image to /fI-file/fP.
```

Как еще разработчику конвертировать десятки страниц, где хранятся сотни опций, из формата вывода в исходный формат? Что типично для проекта Longjump, человеческие и организационные затраты (воистину потрясающая глупость этой работы) были проигнорированы менеджерами, и младшему разработчику (Кимберли) было поручено выполнить все конвертирование вручную.

К тому моменту, когда до меня дошла информация об этом поручении, Кимберли была вне себя. Она работала у нас первый год, который был для нее решающим первым годом в компьютерной сфере, и относилась к тем новичкам, которые жаждут показать свое прилежание и преданность работе. Она была готова пойти на все, чтобы выполнить поручение руководства. Но пока мы сидели вместе, определяя рамки работы, листая страницу за страницей бессмысленные коды ее терминала, она воскликнула: «Я не понимаю, как я вообще когда-нибудь смогу закончить эту работу. Я даже не представляю, с чего начать».

Почему компания не обратилась к нашему поставщику и не настояла на том, чтобы они передали нам оперативные страницы руководства в исходном формате? Возможно, у моего начальника и у меня просто было недостаточно опыта, чтобы твердо настоять на этом. Возможно, никто не заметил этого до подписания контракта, а теперь было уже поздно. Но возможно, поставщик просто почувствовал, что Edom Engineering разрабатывает собственный конкурентоспособный код, и сознательно не предоставил нам исходные файлы. В любом случае, казалось, что мы в тупике. Я решил использовать мои скромные способности программиста, чтобы сделать этот проект доступным человеческому пониманию. Древний формат troff, особенно формат вывода, не поддавался механической обработке. Даже усовершенствованный искусственный интелект, скорее всего, не распознал бы все элементы страницы, такие как заголовок раздела (.SS) и разметку абзацев (.TP).

Но я нашел соотношение взаимодействия между женщиной и машиной, которое сработало, и написал несколько макросов на Emacs Lisp, автоматизирующих все, что можно. Кимберли должна была только найти хотя бы какой-то знакомый элемент – например заголовок раздела или тег разметки, – потом установить Emacs-курсор на него и нажать единственную клавишу, чтобы преобразовать данный абзац в правильный

формат. Когда я показал ей основные движения и клавиши, она просто подскочила в своем кресле, и так радовалась, что обработала первую страницу с нескрываемым удовольствием. Потом уже она мне сказала: «Эти макросы спасли мне жизнь».

Капитуляция

Военные корреспонденты, освещающие любой конфликт, всегда передают на родину истории про шоферов, которые избегают главных дорог и КПП, пользуясь объездными путями. Но порой новый маршрут оказывается опаснее того, которого они старались избежать. То же было и с проектом Longjump.

Скорость выполнения этого проекта заставляла время ускоряться, а не замедлять свой ход, хотя Эйнштейн говорил именно о замедлении. Всего через пару месяцев наши разработчики осознали, что новые компоненты системы не работают вместе и что перенос программного обеспечения идет плохо. Мы не должны были терять время, поскольку могли истечь кровью на любом КПП. Инвесторы настаивали, чтобы мы шли по графику, и поэтому кто-то наверху решил радикально сменить направление.

Вместо того чтобы внедрить RISC-процессор в спроектированную и построенную по условиям потребителя систему, нам пришлось стать важным, но дополнительным распространителем продукции другой успешной фирмы, которая продавала компьютеры на основе этого процессора. Чтобы подстроить компьютерную систему, нужно поскрести само устройство и немного приправить программное обеспечение. А это совсем не то же самое, что быстро смешать и испечь, как первоначально планировали наши «шеф-повара» из команды разработчиков.

Многие из нас пали духом, узнав об этой новой стратегии. Хотя использование продукции другого поставщика и задевало нашу гордость, дело было не только в гордости; у нас было достаточно оснований опасаться. Без нашего секретного соуса наш мастерский реинжиниринг системы с головы до ног потерял бы большинство преимуществ производительности. В каком-то смысле мы предлагали потребителям универсальную компьютерную систему, обратно совместимую с нашими предыдущими системами. Только новая RISC-архитектура могла, как мы надеялись, дать достаточно хорошую производительность, которая оправдала бы наши старания.

Недовольные разговоры слышались на каждом шагу. Сотрудники нижних уровней ощущали, что Джордж не соответствует роли руководителя проекта, что неизбежно должно было привести к объяснению их выбора перед начальством.

Рабочие места были украшены невообразимыми пачками отчетов, идущих по иерархической цепочке от каждого участника проекта к генеральному директору. Каждый отдельный отчет характеризовал проект как вонючую кучу навоза, но по мере его продвижения к высшему руководству резкая оценка постепенно смягчалась. И когда информация достигала генерального директора, он мог с радостью заключить, что у проекта есть потенциал для выращивания благоухающих цветов.

Я упомянул в начале главы, что для того, чтобы выжить, пережив такие проекты, как Longjump, рабочая команда должна постоянно решать краткосрочные задачи, используя свои профессиональные навыки. Пара месяцев работы над таким проектом – и подобные попытки начинают сильно выматывать. Не каждую проблему можно разрешить при помощи Emac-макроса, который я написал для Кимберли.

Разработчики стали предварять или заканчивать описание своих заданий комментариями вроде «Не то чтобы это пойдет на пользу» или «Не понимаю, как это заработает». Но скоро в личных отчетах перестали появляться и комментарии. Делиться ими с соседями по рабочему месту было излишне, потому что они и так были согласны с этим. Но выражать их в письменной форме тоже казалось пустой тратой времени, потому что высшее руководство реагировало только на давление со стороны инвесторов.

Хуже всего было то, что никто из сотрудников не мог предложить лучших условий, чтобы мы успевали по срокам и задачам. Но будучи разработчиками, а не инвесторами, мы чувствовали, что для нас реальнее справиться с задачей, отодвигая сроки и разрабатывая систему, которая бы поддерживала наши уникальные новшества, чем обманывая клиентов разговорами типа «у нас такой же продукт», когда на самом деле его сходство с более ранними продуктами заканчивалась интерфейсом. Однако момент для подобного рода дискуссий был упущен. Мнение разработчиков больше не влияло на ключевые решения компании, хотя до этого момента оно всегда учитывалось.

Прорыв

Добросовестно посещая все собрания по проекту Longjump, я имел возможность слышать весь спектр жалоб разработчиков. После многих недель, которые я высидел за покрытыми пластиком столами в похожих на пустые коробки конференц-залах, я ощутил, что перегружен знанием.

Однажды я все рассказал Алану. Я ожидал, что он отреагирует стандартной в данном случае фразой «Отключись от этого и продолжай ра-

ботать», при этом в голосе у него будет звучать нотка похвалы и сострадания, но ответ Алана потряс меня совершенно.

«Я думаю, – сказал он в своем осторожном стиле, тщательно подбирая каждое слово, – ты должен написать дословно все то, что ты мне сейчас сказал, и разослать всем сотрудникам компании».

Что за странная идея! Нужно сказать, что в компании Edom Engineering с ее свободной культурой корпоративного поведения подчиненные в любой момент могли спокойно высказаться от лица всей компании, и об этом моментально узнавали все, от команды производителей до генерального директора. Такая практика применялась не только в случае официальных дел компании, но и для того, чтобы объявить вечеринку, перенести футбольный матч или рассказать анекдот. Не было ничего пугающего в том, чтобы «прикрываться лицом компании».

Но знаю ли я что-то достойное того, чтобы отнимать у людей время? Все, что я рассказал Алану, было общеизвестно и совпадало с общим настроением команды. Как я уже сказал, разработчики просто перестали говорить об этом, потому что это ни к чему не приводило, и продолжать дискуссию было бесполезно.

И все же мысль о том, чтобы выразить это общее настроение в письменной форме, взволновала меня. Каждый разработчик испытывал те же чувства к проекту Longjump, но изолированно. Они не могли поделиться своими эмоциями во время какого-нибудь психотерапевтического сеанса (хотя у нас в компании был психолог, который, однако, ничем не занимался). Мы все находились в ситуации, которую считали ложной, и день за днем теряли ту крупицу здравого смысла, которая была в нас, пока мы жили правдой. И теперь нас стремительно затягивало в черную дыру с ее невыполнимыми требованиями.

Итак, мысль о том, что все наши огорчения и страхи могут в очередной раз быть высказаны вслух, подняла мне настроение. Думаю, Алан правильно определил, что именно я могу выразить их за всю команду. Не только потому, что я писал, но еще и потому, что был руководителем команды, и это давало мне дополнительную, хотя и плохо заметную точку опоры в обмен на информацию, которую я получал о проекте.

Помню, как он рассуждал о своем предложении, сидя в непринужденной позе нога на ногу. Он убеждал меня, что мне не следует терять хватку. «Ты ясно обрисовал мне сейчас проблемы, стоящие перед всеми нашими коллегами, и я настаиваю, чтобы в письменном виде ты выразил их так же откровенно, прямо и бескомпромиссно».

Этот путь казался опасным, но к этому моменту я уже ничего не боялся. Однако неуверенная мысль о том, как это отразится на моей карье-

ре, меня посетила. Я знал, что меня назначили руководителем рабочей группы за мои достижения в работе, из-за этого даже в разгаре страстей меня не могли сместить с должности или оказывать на меня давление. Беспокоясь о судьбе Алана, я все же спросил: «И все же, как ты поведешь себя, если топ-менеджеры будут недовольны?»

«Не беспокойся, я тебя прикрою; в любом случае, не вижу оснований для отрицательной реакции руководства». Эти слова, сказанные все тем же спокойным голосом, придали мне уверенности.

Даже если бы я сохранил копию электронного письма, которое отправил в тот день, его формат сегодня бы поддерживали очень немногие компьютеры, потому приношу извинения, что не могу его здесь процитировать целиком. Оно было коротким, всего несколько абзацев. Я написал о том, что команда, работающая над проектом Longjump, пришла к выводу, что общее направление работы идет под откос. Я обобщил задачи, которые стояли перед разработчиками на ближайшие несколько недель, и связанные с ними проблемы, такие как сомнения в выполнении задачи в срок, нехватка информации о платформе и разного рода сложности с несовместимостью. Потом я перешел к моральным умозаключениям и закончил тем, что написал о нашей любви к нашей работе, к нашим потребителям и нашей компании, которые все рискуют быть подвергнутыми опасности, если мы будем придерживаться текущего планирования.

Не думаю, что в тот день кто-то успел сделать что-то полезное по проекту. Распечатки моего письма развешивали на стенках и на рабочих местах. Люди поздравляли меня, встречая в коридоре, примерно в таком духе: «Я показал это своей жене, и она спросила, сколько теперь ждать, пока этого человека уволят». Посыпались электронные письма из соседнего штата, где располагалась дружественная нам компания и где меня не так хорошо знали, – в них люди спрашивали, кто же этот смелый и проницательный человек.

На самом же деле, я не думаю, что это история о смелости и проницательности. Я не проявлял смелость, будучи уверен, что меня не тронут. И я не был удивительно проницательным, потому что просто выразил то, что слышал от всех и каждого, и преподнес свои наблюдения в этой точки зрения.

Но вот чем я горжусь по сей день и благодарю Алана за то, что он меня к этому подтолкнул, так это тем, что я выразил и озвучил за всю команду наше общее мнение. Сделав это, я прорвался через ощущение личного страха и паники, которое испытывал каждый в отдельности, и победил вредное для компании ощущение опасности, которым она была охвачена. Выразив за всех то, о чем они думали, я дал всем нам возможность выжить.

В предвкушении менеджмента XXI века

Лишь начав записывать свои воспоминания в виде главы для книги «Идеальные команды», я осознал, что в моем письме заключался более глубокий смысл: я не просто выпустил пар. Думаю, мы с Аланом встали на путь, ведущий нас к новому типу руководства и осознанию роли нового стиля коммуникативных технологий. Этот смысл в течение 20 лет так и оставался только смыслом, без формы, пока консультанты по вопросам управления не озвучили и не оформили его как направление в корпоративном поведении.

В начале главы я отметил, что для управления компании Edom Engineering характерно с уважением относиться к разработчикам и терпеливо реагировать на самоуверенный стиль, которым они выражают свое мнение, – это был девиз нашей компании на заре ее деятельности. Что-то в проекте Longjump заставило этот девиз пошатнуться, возможно, причиной было нежелание генерального директора вести переговоры с инвесторами, нежелание Джорджа спорить с генеральным директором и вместе с тем – новые черты управления, которые появились при слиянии. Разработчики компании не были внутренне готовы организовать себя с целью сохранить за собой право участвовать в принятии решений в сложное время, когда корпоративная культура поведения рушилась. Но на сцену вышел другой фактор, техногенный по своей природе, и предложил альтернативный подход к распределению сил и влияний.

Мы жили тогда в эпоху, когда Интернет еще не был общедоступен, когда доступ к Всемирной паутине был только у десятка сайтов. И все же уже тогда онлайн-сообщества стали проводить эксперименты с политической активностью широких масс, которая стала набирать силу наравне с традиционными лидерами и партийными центрами. Несмотря на то что книга Говарда Рейнгольда «The Virtual Community: Homesteading on the Electronic Frontier» (Addison Wesley) не была опубликована до 1993 года[1], а Джон Барлоу не выпустил еще свою «Декларацию независимости Киберпространства» (которую теперь воспринимают как историческую веху) – она была опубликована только в 1996 году, – знание о виртуальном сообществе WELL, в котором участвовал Рейнгольд, быстро распространялось среди пользователей Интернета, пока я работал в проекте Longjump. И я уже тогда работал над тем, чтобы создать новый подход к совместной работе – когда коллеги, разбросанные по свету, могут сотрудничать посредством электронной почты и форумов,

[1] Позже Алан опубликовал критический отзыв на эту книгу, но этот факт не отменяет мое замечание о том, что он использовал онлайн-общение, понимая его потенциальную силу.

которые поддерживаются компьютерными профессионалами за социальную ответственность.

Я уверен, что смелое предложение Алана распространить мое мнение при помощи электронной почты было ответом на дремлющую силу, которую он видел в цифровых коммуникациях. Алан плохо помнит этот период своей карьеры, поэтому не смог ответить, сознательно ли он предложил использовать потенциал цифровых коммуникаций, пока я невнятно излагал ему свое недовольство сроками и неполадками в его компании, но эффект нашего тайного сговора был в том, чтобы запустить в виртуальное пространство незапланированный эксперимент, направленный на массовое вовлечение людей в некий процесс. Мое стремление немедленно что-то сделать тоже показывает, что я был подготовлен своей политической работой онлайн к тому, чтобы сотворить некое «существо», которое мы спустили с цепи.

Результаты эксперимента оправдали наши усилия и ожидания. Цифровые сети – это мощные средства демократии, по крайней мере, для тех, кто уже связан с Сетью. В той вырождающейся корпоративной среде, к тому же еще испытывающей финансовые затруднения, мы не могли надеяться на чудесный результат. Эффект, заметный по изменению морального состояния сотрудников, – вот лучшее, на что мы могли рассчитывать. Очевидно было, что те же технологии могут реально поддержать конструктивное управление проектами в компании с более здоровым климатом, если применять их на более ранних стадиях и поддерживать просвещенным руководством и социальными нормами.

Представление, которое мы устроили в тот день (бессознательно в моем случае и, возможно, сознательно в случае Алана), теперь мелькает на страницах учебников по менеджменту. Это стало принципом просвещенного менеджмента, когда еще только начали появляться такие книги, как «Манифест пути» Кристофера Лока, Рика Левина и др., книга Эрика фон Хиппеля «The Sources of Innovation» (Oxford University Press), которые акцентировали значение открытости в управлении. Еще далеко не все компании научились извлекать наибольший потенциал из сотрудников (или потребителей и других заинтересованных лиц), который, будучи организован определенным образом, может стать явным и важным фактором принятия решения в организации. Но многие компании говорят об этом, и в конце концов такое поведение будет восприниматься как обычное.

Заключительные замечания

Сотрудники Edom Engineering были не из тех, кто будет ждать, пока водоросли разрастутся и заглушат пруд. В тот день они прочитали мое

электронное письмо и среагировали на него, а потом снова углубились в бездну своей работы.

Ни один менеджер не призвал меня к ответственности за сказанное в письме – вероятно, они очень хорошо знали, что происходит, невзирая на то, был ли Джордж откровенен с ними, – но в то же время направление движения проекта Longjump не изменилось. В конечном счете, учитывая новую модель ведения дел, которая основывалась на перепродаже системы компании-конкурента, мы достигли поставленных перед нами целей, закончив проект вовремя и, формально, с успехом. Наш продукт провалился на рынке.

На самом деле RISC-процессоры так и не дожили до момента, когда их стали активно рекламировать. Признанные производители процессоров сумели повысить производительность при предполагаемом усложнении их устройств, используя такие стратегии, как предварительная выборка инструкций, интеллектуальное планирование и многожильность. Некоторые из этих стратегий, такие как разбивание крупных инструкций на более мелкие с более последовательной структурой, они позаимствовали у RISC-разработчиков.

Эти признанные производители держали низкие цены при повышении эффективности от роста масштабов производства и вкладывали вырученные в результате доминирования на рынке средства в дорогостоящее современное оборудование, которое могло производить процессоры по еще более низкой цене. Производители процессоров RISC так и не смогли сориентироваться. Итак, несмотря на то что производители RISC-процессоров остались на рынке, RISC как новый элемент на рынке электроники оказался плохо просчитанным ходом большой группы специалистов и инвесторов в компьютерной индустрии. Последняя и большая надежда компании Edom Engineering оказалась просто сноской в этой истории.

Меня так и не уволили. Я проработал в компании еще несколько месяцев, до того момента, пока проект по составлению документации, в котором я участвовал (безопасная многозадачная библиотека Motif), не закрыли.

Когда это случилось, я пошел к начальнику моего начальника, возглавлявшему большую команду разработчиков, с которым у меня были отличные отношения, и сказал ему: «Главный проект, над которым я работал, закрыли, и у меня мало работы – как думаешь, чем бы я мог заняться?»

Разумеется, это расшифровывалось так: *Я не вижу никакого будущего для своей карьеры в этой компании, и мне нужно, чтобы ты привел аргументы в пользу того, чтобы я остался.*

Начальник моего начальника улыбнулся своей неизменной улыбкой и ответил: «Не беспокойся об этом. Найди время и займись учебой».

Что можно было расшифровать так: *Ничего не могу тебе предложить. Так что вперед – ищи новую работу.*

Мой уход из компании был частным случаем массового исхода сотрудников из компании в момент ее головокружительного, но закономерного упадка. Поэтому я почти не следил за чередой «приобретений» другими слабыми компаниями в нашей области. Теперь я чувствую, что было бы лучше для всех – и для сотрудников, и для инвесторов, и для потребителей, – если бы этот массовый исход случился до начала проекта Longjump. Никому не помогла эта попытка спасти умирающий проект умирающей модели предприятия.

Но поскольку все мы решили принять участие (из любви к тем продуктам, которые разрабатывали с таким тщанием многие годы) в этом последнем рывке, предпринятом созданной нами компанией, мы по крайней мере смогли выиграть в смысле моделей поведения в экстремальных условиях, которые выработали во время работы. Не могу утверждать, что мое письмо было правильным способом выразить недовольство и страх, которыми я пропитался, но уверен, что это была тактика психологического выживания, которая помогла многим моим коллегам. В этом свете интересно, что оно стало еле узнаваемым предвестником новой модели открытой передачи информации при помощи современных цифровых сетей, которые в свою очередь очерчивают новый тип устройства в корпорации, организованный снизу вверх, где учитывается мнение сотрудников, – именно тот демократический тип корпорации, который продвигают сегодня бизнес-эксперты.

V

Музыка

Эту книгу мы начали с интервью с Тимом О'Рейли, потому что считали, что в нем отражено множество тем, в которые мы погрузимся в дальнейшем. В частности, Тим провел параллель между совместной работой разработчиков и музыкальных групп. Нас это заинтересовало, и мы решили копнуть поглубже.

Тим: Должен сказать, что, по правде, моя самая большая слабость как лидера команды – всегда брать то, что доступно, пытаясь что-то из этого извлечь, а не инженерный подход «нам нужно то, то и то», когда что-то создается из ничего.

Дженни: **Об этом и мой вопрос. Одна из ваших фраз заставила меня задуматься: традиционный путь решения того, что вы хотите создать, – добиться общего согласия заранее. Мне же кажется, что вы больше заинтересованы в процессе поиска, чем в том, что именно получится в результате. Так ли это?**

Тим: Что ж, и да и нет. Хороший вопрос. Он парадоксален, потому что вы как бы говорите, что любая истина получается из опыта, – и это звучит, как нью-эйдж в Калифорнии. Когда я говорю «нужно увидеть статую в камне», она на самом деле там. Единственная. Ладно, может, их там и больше, но в любом случае это нечто новое. Нельзя больше просто делать что угодно, плывя по течению. Процессом управляет чувство прекрасного.

Дженни: **Значит, лидер должен обладать подобным эстетическим чувством?**

Тим: Вот именно. Нужно знать, к чему стремишься. И здесь мы опять возвращаемся к тому, что нужно напеть мелодию и убедить других следовать за тобой. Это идея гармонии. Видите ли, мелодия общая, и здесь нельзя фальшивить: нельзя, чтобы все играли что хотят. Мелодия общая. Ее нужно найти, слить воедино. В этом и состоит искусство сплочения креативной команды.

Возвращаемся к музыкальной аналогии. Пусть люди сами учатся играть на своих инструментах: если они серьезные музыканты, то могут это сделать. Тогда они смогут улучшать мелодию на основе уже существующей. Но это начинается с того, что Эрик Реймонд в мире программирования с открытым исходным кодом называет «убедительным обещанием». По чисто эстетическим причинам люди могут сказать: «Что ж, я бы хотел в этом участвовать». Они могут проявить собственные творческие способности. И если вы все делаете правильно, то не предъявляете лишних требований, не говорите, что или как делать. Вы просто показываете им конечную цель, а уж путь к ней они найдут сами.

Эндрю: **Хорошо, но у меня вопрос. Есть очень классные, выдающиеся лидеры групп. Например, Каунт Бейси, который издает не так много альбомов и редко участвует в чартах, как и его музыканты. Но я уверен, что в истории немало не ставших популярными групп, которые хотели работать так же, но не смогли. И то же самое мы видим в программных проектах. И сотрудники талантливы, и лидер умен и талантлив, а с разработкой у команды все равно проблемы.**

Тим: За свою карьеру я рассказал множество подобных историй, чтобы помочь кому-то организовать свою деятельность. И сам я как раз совсем не авторитарный лидер или менеджер. На самом деле я часто шучу, что похож на главного героя мюзикла 1950-х «Продавец музыки». Знаете, о чем он? Тот парень был мошенник, он торговал инструментами и как-то сказал, что будет учить детей музыке. Но о музыке он, конечно, ничего не знал. Он заявил, что у него есть новая система – «Система мысли»: он напоет мелодию, и дети сразу поймут, как сыграть ее на своих инструментах. В мюзикле это, естественно, сработало: дети действительно как-то все поняли, и очень кстати, потому что он как раз влюбился в городскую библиотекаршу.

Эндрю: **Можно ли так руководить группой на самом деле?**

Тим: Конечно, если ваши музыканты очень-очень квалифицированные. Собственно, в этом и заключается джаз.

Эндрю: **Видимо, вы правы, Каунт Бейси так и делает.**

Тим: Да, джаз-банд – это «Система мысли» в действии. Кто-то находит идею, а остальные ее подхватывают. Я до сих пор вспоминаю, как мы с Джеффом Безосом обсуждали патентную систему с членами Конгресса. Одним из конгрессменов был очень известный человек; мы пытались проводить аналогию между процессом изобретения и джазом, и он все отлично понял, потому что очень любил джаз: «О Господи! А если бы все джазовые риффы были запатентованы, и их нельзя было бы брать и играть?»

Так что, думаю, во многом аналогия действенна. Если у вас достаточно хорошие сотрудники – мотивированные, активные и умелые, можно наиграть им мелодию, и они будут импровизировать на этот мотив. Именно этим я и занимаюсь всю свою карьеру. Это не означает, что практики остается мало. Но даже практическое обучение состоит в рассказывании историй. Например, есть команда из редактора и автора, и я как редактор часто беру и переписываю какой-то фрагмент, после чего мне говорят: «Да-да, теперь я понял». Точно так же я как редактор работаю со своими помощниками: я демонстрирую им что-либо, а они демонстрируют, что именно поняли из моего показа. И когда я убеждаюсь, что они поняли, мне уже мало что остается сделать.

Понимаю, что сейчас высказал две противоречащие друг другу идеи: «увидеть статую в камне» – и «определить цель, в которую поверили бы другие». Первая состоит в том, что надо найти идеальную форму, притом единственную; согласно другой можно сделать все из всего. А истина, как обычно, где-то посередине. Изнутри этот процесс, если все сделано правильно, подобен обретению гармонии в музыке. Возможны несколько вариантов гармонии, но нужно остановиться на чем-то одном.

Так что в каком-то смысле эта аналогия вполне работает. Чтобы ситуация разрешалась, есть несколько причин, и они должны сочетаться. Большую роль в моем мышлении играет общая семантика Альфреда Кожибского с ее тезисом: «Карта – это не территория». Карта соответствует территории. И когда вы порождаете эстетическую цель, которая каким-то образом соответствует территории, делая вещь большей, чем она есть, то вам удается убедить людей идти за вами. И когда вы нащупали этот момент, смогли сформулировать цель, люди соглашаются с ней. А вместе вы создаете что-то подлинное.

Редактируя наше интервью с Тимом, мы сделали его независимым разделом, потому что поняли его важность. Мы увидели здесь проницательное утверждение о том, что команда может собраться вокруг сильного лидера, и что лидер, видя конечную цель, может объединить всех вокруг себя. Но тут Тим сказал нечто еще более поразительное: он фактически стер грань между целью проекта и художественной целью.

Мы решили, что эти идеи нужно изучить подробнее. Мы стали искать специалиста, который рассказал бы нам о том, как работают музыканты, и с радостью узнали, что Тони Висконти готов уделить нам время. Мы оба много лет были большими поклонниками его работ, даже не зная, кто стоит за этой музыкой. Всю свою карьеру Тони провел, придумывая, как заставить группы – команды, если хотите, – музыкантов работать вместе.

Мы не совсем представляли, что из этого выйдет. И нас сильно удивило, что многое из сказанного Тони пересекалось с тем, что говорили в этой книге другие наши участники. Читая это интервью, вспомните, что уже говорилось: постарайтесь сравнить идеи, ощущения и мнения в этой главе с тем, что вы читали на протяжении предыдущих нескольких сотен страниц.

31

Создавая музыку

Интервью с Тони Висконти

Если вы все еще не знаете, кто такой Тони Висконти, просто посетите страницу сайта (http://www.tonyvisconti.com) и щелкните на ссылке Discography. Если вам не меньше сорока, то у вас почти наверняка есть хотя бы несколько альбомов, которые он выпустил (включая и ваши любимые). Он работал со звездами, великими представителями музыкальной индустрии, и он создавал альбомы, которые имели как творческий, так и коммерческий успех. Мы бы хотели узнать, как он это делал... и к нашему потрясению, он спокойно сел рядом с Эндрю и все нам рассказал! И все, что он сказал, неожиданно оказалось важным для разработки программного обеспечения.

Эндрю: **Большое спасибо за то, что нашли время побеседовать с нами. Поскольку вы продюсер, я бы хотел начать с разговора о том, чем вы занимаетесь. Потому что, пожалуй, мало кто представляет, чем занимаются продюсеры. Вспоминается эпизод из «Приключений Рокки и Бульвинкля», в котором Борис Баденов нанимает Бульвинкля в качестве кинопродюсера. Бульвинкль спрашивает у Бориса, что же делает продюсер, и Борис отвечает: «Деньги». Уверен, что у профессий «продюсер», «арт-директор», «издатель» и «руководитель проекта» много общего. Итак, как бы вы описали то, чем занимаетесь?**

Тони: Что ж, определения этих профессий очень туманны, это в лучшем случае. Но если кто-то делает какой-то вклад в некий вид деятельности, этот вклад надо как-то определить. Мой стиль работы отличается от того, как это делает например Хью Пэдхам (Hugh Padgham) или даже Джордж Мартин. Продюсер звукозаписи – это что-то вроде, ну, скажем, менеджера по инвестициям. Теперь мы уже не собираем деньги, но раньше продюсеры занимались поиском средств. Сегодня продюсером альбома или проекта по сути является фирма звукозаписи.

Некоторые говорят: «У нас есть деньги, запишите нам такой-то альбом». Таким образом, эти они сразу же определяют род нашей деятельности. Я, в свою очередь, как режиссер звукозаписи полностью управляю созданием записи с того момента, как она попала ко мне, и до момента, когда она попадает на прилавки. Вся ответственность – на мне.

Эндрю: **Звучит очень похоже на то, чем занимаются руководители проектов. С тем, чем занимается руководитель команды, тоже много общего. А когда этим занимается ведущий разработчик, он на самом деле берет на себя функции руководителя проекта.**

Тони: Да, думаю, именно этим и занимается руководитель проекта – и, возможно, в вашей сфере это примерно то же самое.

Эндрю: **И то, что вы сказали о туманности определения таких профессий, однозначно имеет смысл в нашем случае. Спросите любого программиста в команде: «Чем занимается руководитель проекта?» И если они вообще смогут ответить на этот вопрос, то их ответ будет примерно таким же, как ваш. Подозреваю, что если спросить у музыкантов какой-нибудь группы, записывающей альбом, чем же занимается продюсер, им также будет трудно ответить на этот вопрос. С той только разницей, что музыкантам он иногда говорит, что надо делать, и постоянно не соглашается с ними.**

Как вы думаете, чем именно определяется, является ли кто-то хорошим продюсером или эффективным руководителем проекта?

Тони: Я думаю, что тот, кто собирается стать руководителем проекта, должен обладать в первую очередь глубоким знанием о той области деятельности, которой он будет руководить. А именно, знать специфику каждого этапа работы, понимать разные уровни работы, которую надо выполнить; например отличать творческую команду от практической. Полагаю, есть члены команды, отвечающие за креатив, и те, кто отвечает за выполнение, чтобы все было сделано как следует. То есть продюсер должен во всем этом разбираться.

Думаю, что продюсер или руководитель проекта должен иметь высокое социальное происхождение и практический опыт по всем родам деятельности, которыми он собирается руководить. В противном случае он не будет по-настоящему владеть ситуацией и станет заложником того, кто захочет выйти сухим из воды, совершив преступление, в нашем случае – того, кто попытается свести свои обязанности к минимуму. Я сам прошел через все профессии, связанные с музыкальным бизнесом. Меня назначали отвественным за звукозапись. Я играл на бас-гитаре, на гитаре и на пианино. В какой-то момент я освоил профессию режиссера звукозаписи. Потом научился делать окрестровки и писать партитуры – да, я писал настоящую музыку на настоящей бумаге, и музыканты читали ее с листа.

Сегодня я сам уже не делаю все это на том же уровне, потому что раньше каждое из перечисленных занятий было для меня отдельной профессией. Теперь я могу иногда побыть бас-гитаристом, а иногда – оркестровщиком. Я могу справиться с большой командой в момент звукозаписи, потому что точно знаю, что должен делать каждый из них.

Эндрю: **Но при этом вам ведь необязательно быть лучшим в каждой из этих профессий? Лучшим музыкантом, или лучшим оркестровщиком, или лучшим звукорежиссером среди всех, с кем вы работаете? Не нужно быть лучшим – важно просто иметь практический опыт в каждой из этих профессий?**

Тони: С течением времени я ловлю себя на том, что мне не хочется заниматься одновременно всеми профессиями, просто потому что это слишком сложно – заниматься ими одновременно. Но когда у вас маленький бюджет, каждому приходится становиться мастером на все руки. К примеру, если бюджет настолько мал, что я могу позволить себе только одного ассистента, а нам надо записать музыкальную группу, то я параллельно исполняю сразу несколько обязанностей. И в этой ситуации я, в каком-то смысле, управляю самим собой. Но в идеале хочется работать с командой профессионалов. Я знаю, что я хороший бас-гитарист, но я хочу работать с потрясающим бас-гитаристом.

Я знаю, чего мне ожидать от каждого члена моей команды, потому что у меня есть личный опыт. Я знаю возможности каждой профессии, и очень часто мои требования граничат с невозможным. Я могу добиться хорошего исполнения от музыканта только потому, что знаю, просто знаю, что он может сделать. И для этого мне необязательно быть лучше их, но, думаю, у меня достаточно оснований направлять их и давать им советы просто потому, что я раньше занимался тем, чем теперь занимаются они.

Эндрю: **Похоже, тут почти та же история, что и с уважением. Кто-то по-настоящему прислушивается к твоему совету или требованию только если он тебя уважает. По крайней мере уважает твое знание о его профессии.**

Тони: Да, это сродни боевому искусству. Ты не будешь уважать кого-то, кто занимался боевым искусством только шесть месяцев, а теперь собирается тебя учить. Даже как новичок ты видишь, что его тело плохо скоординировано, и он просто не может показать то, о чем говорит. Но когда тебя обучает большой мастер боевого искусства, который занимается этим уже 35 лет, то появляется уважение, и ты готов сделать все, что он тебе скажет. И если такой человек действительно сам пережил все, о чем говорит, ты просто не можешь не уважать его. А он будет уважать тебя, потому что ты пришел к нему, чтобы чего-то достичь, чему-то научиться.

Вот так я себе это представляю. Не знаю, может, это звучит высокомерно, но я прошел через все это достаточно давно, потому что стал заниматься музыкой с очень раннего возраста. Я бы сказал, что к 25 годам уже прошел все необходимые этапы пути к профессии режиссера звукозаписи. Единственное, чего мне не хватало, – это умения общаться с людьми. Пока я осваивал этот навык, я набил много шишек, учась на своих ошибках. Уметь говорить с людьми на самом деле очень важно. По-настоящему общаться, без раздражения, без злости, при этом в голосе не должно быть фальши, лицемерия.

Эндрю: **Что случится, если плохо общаться с людьми, говорить с ними неискренне и неуважительно? Проявлять лицемерие? Что в этом случае пойдет не так с вашей командой или проектом?**

Тони: Ну, если вы не будете решать возникающие проблемы открыто и эффективно, то потеряете заинтересованность команды. Они перестанут вас уважать, утратят интерес к вашим словам. Они будут думать, что вы сами не знаете, о чем говорите. Худшее, что может случиться с продюсером или с руководителем проекта, это когда команда воспринимает его как какого-то придурка без понятия. Не доводите до этого, потому что навсегда потеряете свою команду.

И вот что я вам еще скажу. Если вдруг так случилось, что вы сплоховали, сделали что-то не так, единственный способ поправить положение, это честно сказать команде: «Я совершил ошибку, давайте попробуем еще раз. Я все обдумал и понял, что это было неправильно». Вот таким вот образом можно вернуть уважение команды после того, как вы его потеряли, – признать, что вы допустили ошибку. Честно и смело признавая свои ошибки, вы удерживаете уважение команды. Выкручиваясь, скрывая свое невежество или постоянно пытаясь выглядеть крутым, вы все сильнее увязаете в своем обмане, и ваша команда просто уйдет от вас.

Эндрю: **Я совершенно ясно прослеживаю параллели с руководством группой программистов. Уважения, вызываемого руководителем группы, недостаточно. Он еще должен обладать определенной простотой (хотя мне не нравится это слово), по крайней мере готовностью признать, что кто-то кроме него может быть прав.**

Тони: Да.

Эндрю: **И я знаю много программистов, у которых с этим сложности. Думаю, то же самое можно сказать и о рок-звездах.**

Тони: Мнения часто различны, но иерархия всегда одна. Тот, кто управляет проектом, держит бразды правления. Это всегда должно быть четко, и это надо понимать. Но это не дает вам право становиться тираном. Одна из самых правильных целей для хорошего руководителя проектом – это дать возможность каждому, кто работает над проектом, максимально проявить себя, не забивать их своим авторитетом, не тормозить их, не загонять их в рамки. Очень опасно допускать такие вещи. Что вам надо делать, так это создавать героев в своей команде, давать людям возможность себя проявить, и, по возможности, сделать так, чтобы каждый из них блистал, – это действительно сделает вашу команду первоклассной.

И вы не добьетесь этого, если не будете их слушать. Для того чтобы они стали членами вашей первоклассной команды, вам надо сделать так, чтобы их мнение было услышано, вы просто должны, обязаны этого добиться. Это также означает, что вы – гибкий и думающий лидер, способный корректировать цели проекта. Если вдруг кому-то в голову приходит шальная идея, и вы видите, что ее можно вставить в проект (а ведь нередко озарения посещают именно рядовых членов команды), вы должны быть готовы пойти навстречу таким проявлениям. Если даже на первый взгляд эта идея кажется вам нелепой, не делайте поспешных выводов. Не говорите «О нет, этого нет в проекте, так что это не имеет смысла». Подумайте немного, дайте себе время осознать новые идеи.

Если подобная шальная идея окажется плохой, не унижайте того, кто ее внес. В этом случае вы должны донести до команды мысль, что это просто идея, а идеи могут приходить в голову спонтанно. Даже отклоняя чью-то идею, вы не должны переходить на уровень выяснения личных отношений. Вы можете сказать, например, что эта идея была хорошей, просто она не очень вписывается в текущий проект. Есть такой способ общения, когда вы стараетесь выявить лучшее, что есть в сотруднике, – нет ничего плохого в том, чтобы осознать, что в данный момент он выбрал ошибочное напрваление.

Эндрю: **А если вы ощущаете, что весь проект движется в ошибочном направлении? Вам нужно вернуть его в истинное русло, но есть ощущение, что многих в команде будет сложно убедить в своей правоте. Даже если они будут довольны конечным продуктом, уже после того как направление было изменено, сама идея изменения плохо воспринимается их сознанием. Как вы справляетесь, когда кто-то не хочет что-то делать, например когда изменения, в необходимости которых вы твердо уверены, встречают отпор? Вам приходится сообщать и плохие новости. Как вы справляетесь с такой ситуацией?**

Тони: Ну, специфика моего дела в том, что многое в нем совершается интуитивно. Музыка – эмоциональный продукт. Под музыку люди смеются и плачут, делают детей и танцуют. Музыка выполняет очень много функций в жизни, и она очень, очень эмоциональна. И я думаю, что каждый в процессе создания музыки в какой-то степени переживает похожие ощущения. Вы как будто ощущаете нутром, что все идет в нужном направлении. Я имею в виду то, что после прослушивания предельно ясно, если кто-то действительно очень здорово сыграл, – в таких случаях он вступает, и люди улыбаются. Если вы не видите улыбающиеся лица, да и сами не улыбаетесь, то это означает: что-то не так. Думаю, в таких случаях руководителю проекта просто надо сказать: «Мы идем не туда. Нужно вернуться к началу и все переделать».

И не надо сокрушаться по поводу потерянного времени, поскольку важно то, что команда вместе сошла с дистанции и теперь вместе же должна вернуться в правильную колею. Я так это ощущаю. Однажды я всю ночь записывал одну композицию, кажется вокальную, не помню. На следующий день мы пришли в студию и думали, что запись будет хорошей – ведь мы всю ночь над ней работали. Но надо всегда смотреть правде в глаза – на самом ли деле все так хорошо? Я считаю, что такие моменты для команды – это общий опыт, который мы приобретаем вместе. Такие тяжелые моменты легче переживаются после того, как все отоспятся.

Все просто: что-то сделано либо хорошо, либо плохо, и кто-то должен сказать об этом. Я твердо верю в честность в такой ситуации. Когда та-

кое случается, как мне кажется, люди по-настоящему начинают понимать, чем они занимаются.

Эндрю: **Вам знакома ситуация, когда вы ощущаете единение команды, но при этом конечный продукт не настолько хорош, как хотелось бы? Все довольны тем, как они справились с задачей, но при этом кто-то платит за это, например – фирма звукозаписи, и когда вы с гордостью приносите им запись на прослушивание – не важно кому именно, – вы можете по их лицам прочитать, что это совсем не то, чего они ожидали. И совершенно ясно, что они не собираются продавать этот альбом.**

Тони: Да.

Эндрю: **Как вы справляетесь с этим? Скажем, к примеру, они просят вас забрать альбом обратно и начать все заново, дав вам на это пару месяцев и хорошее финансирование. Но вы знаете и то, что ваша команда просто не захочет даже слышать об этом. Никому не хочется возвращаться и начинать все с нуля, особенно если они считают, что все сделано хорошо. Что тогда происходит? Как вы поступаете?**

Тони: Часто случается так, что вы долго ждете, пока тот, кто спонсирует проект, сможет его прослушать. Откладывая работу до последнего, вы сильно рискуете. Заказчика не волнуют внутренние сложности вашей команды. Для него это не имеет значения. Если ваш конечный продукт его не устраивает, он скажет вам: «И вот это вы мне приносите? У вас же было столько времени! И за *это* я платил вам?» Вот вам и сложная ситуация. Каким-то образом вам надо убедить вашего работодателя, что результат просто «другой», немного не тот, который они ожидали услышать, а своей команде сказать, что необходимо кое-что доделать и поменять.

Некоторые альбомы, которые мне пришлось представлять, были забракованы или изданы с большой неохотой. Иногда ты просто опережаешь время. Один музыкант, продюсером которого я был, в молодости опережал время года на три. Он не совпадал с ожиданиями публики. В таких случаях вам приходится представлять что-то, по вашему мнению, лучшее из сделанного до сих пор, но когда альбом выпускают, единственная реакция, которую вы получаете, – это пустые взгляды. Проблема, однако, была в том, что мой музыкант был необычным. С одной стороны, он не плыл против течения, а с другой – не шел на поводу у публики.

Я нашел два способа избежать большого непонимания в конце работы. Когда проект уже наполовину или на три четверти готов, я приношу его в заказчику и спрашиваю: «Что вы об этом думаете? Мы на правильном пути?» Не всем музыкантам это нравится, но на самом деле это очень важно сделать, пока вас еще не совсем далеко унесло. не нужно считать тех, кто играет для всех, врагами.

С точки зрения психологии, на это нужно много времени, потому что люди, которые вкладываются в проект, ощущают себя причастными к нему с самого начала, будут более расположены к окончательному результату, если не забывать о них и в разгар работы. Раньше я не любил это делать. Мне нравилось ждать окончания, потому что я был тогда довольно молод и самолюбив. Но на самом деле, каждому нужна обратная связь, и эта связь идет в двух направлениях, потому что вы говорите заказчику: «Я уважаю ваше мнение и хочу, чтобы вы высказали то, что думаете о проекте, на этом этапе его готовности». Это один из способов избежать столкновения авторитетов по окончании проекта.

Эндрю: **Мне очень нравится. Идея в том, чтобы представить работу как поэтапный процесс и работать над какой-то частью проекта, пока не почувствуешь определенную завершенность. Ощутив, что достигнут какого-то промежуточный итог, приносишь это заказчику и говоришь: «Вот что у нас есть на данный момент. Посмотрите, движемся ли мы в правильном направлении?» После чего заказчик «помечает» эту часть проекта, ощутив причастность к окончательному продукту. Проделав это несколько раз за проект, можно сконцентрироваться на том, по поводу чего не возникает разногласий.**

Тони: Интересно, ваши программисты так же относятся к работе, как и рок-звезды, с которыми мне приходилось работать? Они считают, что в фирмах звукозаписи работают глупые и корыстные люди, не представляющие, как должна звучать хитовая запись, даже если услышат ее. Но режиссер звукозаписи (или руководитель проекта) должен соединить эти два мира. Фирма звукозаписи ответственна за все «приземленные» расходы, за материальное обеспечение, в то время как программисты или музыканты отвечают за культуру, новизну, искусство и творчество. Руководитель проекта должен фильтровать все, чем обмениваются эти два мира, потому что если бы программисты напрямую общались с фирмой звукозаписи, их бы спустили с лестницы. И тут же уволили за дерзость.

А если представителям фирмы звукозаписи позволить общаться напрямую с программистами (или музыкантами), то ситуация повторится с точностью до наоборот. Вам же надо понимать, как работают мозги у людей из этих двух миров. Разница примерно как между правым и левым полушариями. И как раз здесь руководитель проекта может блеснуть своим талантом работы с людьми. Как я уже говорил, вам нужны технические умения. Вы должны понимать проблемы, с которыми сталкиваются программисты. Но с другой стороны, вы должны понимать и «приземленные» проблемы компании, и то, *что* они хотят продать покупателю. Вот что мы имеем в итоге.

Второй способ, который я применяю: планирую расходы больше, чем реально необходимо. Это моя маленькая хитрость. Я также прошу несколько лишних недель срока, которые мне скорее всего не пригодятся, а потом сдаю проект раньше срока и с экономией бюджета.

Потому что в глубине души я знаю, сколько реально потребуется времени на проект, – но это знание, безусловно, приходит только с опытом. В этом случае, если вам вдруг понадобится несколько лишних недель, они у вас будут, и все могут дышать свободнее. Если вы запланируете все, исходя только из реального времени, необходимого на проект, у вас не будет потом возможности перейти к Плану Б. И вы сами себя обрекаете на провал. Если вы все-таки сталкиваетесь с тяжелой ситуацией, когда клиенту не нравится ваш продукт, и у вас закончились средства и вышли все сроки – вы проиграли, и всем будет плохо.

Эндрю: **А если члены команды не находят общий язык? Что вы делаете, если между ними возникают конфликты? Если каждый хочет все делать по-своему? Если у вас возникли непреодолимые разногласия, а время между тем идет, и бюджет тает?**

Тони: Ну, если они не находят общий язык, на это надо обращать внимание с самого начала. Я очень тщательно подбираю членов команды. В моей записной книжке множество людей, с которыми я работаю. Такой-то ударник идеально работает с таким-то вокалистом – я просто знаю это. Например, в проекте, над которым я сейчас работаю, заказчик изначально вписал ударника, который, как бы это сказать... не хочу называть имен, но он любит выпить, повеселиться, а еще постоянно рассказывает сальные анекдоты. Он замечательный ударник и вообще отличный музыкант. Но вокалистка – полная противоположность – очень утонченная, не переносит грубость. С самого начала было понятно, что безумие ставить таких людей работать вместе в проекте. Эти отношения постоянно бы тормозили ход работы. Таким образом, получалось, что этот ударник для проекта не подходит, хотя он мне очень нравился. Если бы я его оставил, сеансы звукозаписи превратились бы в пьяные посиделки, а исполнитель, которого я сейчас записываю, – очень чувствительное существо женского пола.

Я стараюсь подбирать членов команды по личным качествам. Это очень, очень важно. Эти качества должны совпадать и подходить друг другу, тогда и со стилем будет все в порядке. При этом совсем не обязательно, чтобы у всех был один и тот же стиль, они могут быть разными, но должны взаимодополнять друг друга. Можно назвать это ин-янь. Может, это правильно. А может и нет. Может быть, не стоит совмещать крайности, противоположные личности и разные творческие начала.

Но это надо каждый раз выяснять, и это ваша задача, если вы руководитель проекта. Узнайте как можно больше о тех, кого вы собираетесь позвать в свою команду. Встретьтесь с ними. Пообедайте вместе. Поужинайте. Сходите погулять. Назначьте свидание. Узнайте этих людей ближе. Если вы будете с ними работать в одном проекте, вам придется иметь со всеми очень близкие, доверительные отношения, и вам не нужны неожиданности. Если кто-то из них может «взорваться» и если я это заранее знаю, то буду готов к этому. Но если эта черта характера вдруг выявится в разгар проекта, и я останусь один на один со звездой, переигрывать будет уже поздновато. Это очень неуютная ситуация.

Должен ли я уволить такого человека? Или я должен поговорить с ним, прочитать ему мораль? Важно то, что я должен о таких возможностях знать до того, как все произойдет. Вы должны стараться собрать свою команду, основываясь на ощущении, что эти люди гармонируют друг с другом. Если вдруг что-то выйдет из-под контроля, у вас всегда остается крайняя мера – уволить этого члена команды. И это совсем неплохо. Это может быть даже очень хорошо, потому что вы освобождаетесь от того, кто доставляет вам много хлопот, и укрепляете команду. Они будут знать, что у вас хватает смелости на такие поступки, кроме того, скорее всего «жертва» увольнения была несимпатична команде. А тот, кто придет на смену и окажется в роли «опоздавшего», будет более сговорчив, стараясь сделать все возможное, чтобы влиться в команду.

Вы видите, как сложно работать, если у вас в команде яркие и разные люди, но если быть морально готовым к самому плохому, вы справитесь с ситуацией лучше, на порядок лучше.

Эндрю: **Увольнять людей – об этом мало кто любит говорить.**

Тони: Знаете, что я скажу? Увольнять людей – это неизбежность. И вы должны допускать такую возможность.

Эндрю: **У меня как-то был начальник, который любил говорить, что единственный способ добиться уважения команды – это четко продемонстрировать, что ты начальник. Мне кажется, он считал, что утверждать свое положение надо, крича на подчиненных, пока они не сделают то, что он захочет.**

Тони: Вы не имеете права кричать на людей. Если вы кричите, они не слышат слов, они слышат только раздражение. Вы должны уметь контролировать это раздражение, хотя это совсем непросто. Вы можете раздражаться, когда говорите правду, но не имеете права показывать свое раздражение. И вам надо научиться в таких ситуациях сделать глубокий вдох, выйти из комнаты, а потом вернуться, уже немного успокоившись, и только тогда договорить правду. Вы не можете орать на людей –

это не работает. Это только будит в них воспоминания о родителях или об учителях и обидчиках. Вы не должны себе это позволять.

Эндрю: **Но бывает же в вашей практике, когда мнения не совпадают, а решение надо принять? Как вы поступаете в таких случаях? Что, если вы уверены, что знаете правильный путь, но при этом кто-то точно не захочет идти по нему с вами? И что вы делаете, когда уверены, что продолжать работу по определенному сценарию нельзя, а тот, с кем вы работаете, не понимает этого?**

Тони: Это щекотливая ситуация, потому что, например, мне иногда приходится отступить, вспомнив, что на обложке альбома имя исполнителя будет напечатано более крупными буквами, чем мое. В моей команде всегда есть «звезда». Иногда я чувствую, что надо что-то изменить. Признаюсь, что когда появляется какая-то новая идея, я прямо заявляю, что не думал об этом и что теперь мне надо серьезно подумать. Потом я говорю: «Может быть, это и неплохая идея. Но я могу и ошибаться».

Таким образом, я всегда оставляю возможность высказаться, но если я совершенно уверен, что проект катится под уклон, то должен напомнить всем о том, что любой проект начинается с определения цели. У вас тоже есть такие собрания на начальных этапах разработки проекта. И все, к чему вы приходите, должно быть записано на бумаге, в таблицах, планах – все это надо создать. Нам нужно закончить то-то и то-то к такому-то числу; в таком случае, заставить человека отойти от заданного плана на финальной стадии работы над проектом – это нарушение договоренности. Именно это и делает команду прочной. Я говорю о социальном контракте, договоренности выполнить определенную совместную работу. И записав все это, вы сможете указать человеку, что он отошел от выработанных совместно правил.

Это очень, очень важно, и обычно это отрезвляет человека, даже «звезду». Но я стараюсь быть на высоте и видеть разницу между легкомысленной идеей и блестящей идеей. И конечно же, Эндрю, если внесена гениальная идея, я непременно говорю: «Жаль, что эта идея никому не пришла в голову с самого начала. Теперь нам нужно уделить ей время». Но вы не можете позволить себе поддерживать несерьезные идеи и предложения. Это даже не обсуждается. Вы должны быть очень тверды, решительны.

Я человек таблиц и графиков. У меня они развешаны по стенам. В данный момент у меня в студии есть график, куда внесены все песни, которые мы записываем. По-горизонтали записаны песни, а по вертикали написано, над чем надо работать в каждой песне. Каждый раз, когда какой-либо пункт выполнен, я его отмечаю – приклеиваю самоклеящу-

юся звездочку любого цвета на этот пункт. Это превращается в своего рода соревнование, потому что исполнитель всегда хочет золотые звездочки (рис. 31.1). Получается что-то вроде игры, поэтому я поощряю всех записывать свои идеи в график. Любой может черкнуть туда свою идею. Это вовлекает в процесс, позволяя всем чувствовать себя творчески и – что более ценно – постоянно быть на виду.

Не только я – каждый в любой момент может посмотреть, на какой стадии выполнения проекта мы находимся. Думаю, это касается каждого – знать, какими темпами мы движемся. И мои графики очень хороши для этой цели. Другой график представляет собой специальную таблицу, в которой расписана вся работа над проектом, с первого по последний день, и там идет обратный отсчет, где последний день идет под номером один, предпоследний – под номером два, так что каждый может наглядно увидеть, на каком дне работы мы находимся. «О, у нас осталось 45 дней. Уже только 42. Три дня до сдачи проекта». И это неприкрытая правда о сроках.

Я делаю так, потому что нахожусь в реальной комнате с реальными стенами. А можно сделать и постоянно обновляемую интерактивную таблицу у себя на компьютерном рабочем столе, наладив ежедневную рассылку всем членам рабочей группы, с которой они могли бы сверяться несколько раз в день. Мы сверяемся с графиком почти ежечасно. Если, например, вы работаете над видеоигрой, то можно услышать такие реплики: «О, посмотри, Джимми приделал ноги персонажу, а Сью записала музыкальную тему. Хорошо, тогда теперь я знаю, что мне надо делать». Я по-настоящему верю, что такой «публичный» график хода работ держит всех в ответственности за проект.

Эндрю: **Получается, этот график на стене – что-то вроде ежедневного средства связи. Все ли участвуют в составлении этого графика? Как устроено общение в команде? С командой говорите вы – или члены команды говорят друг с другом?**

Тони: График выполнения работ появляется во время наших встреч, на которых мы планируем проект, мы вписываем в него все идеи, которые у нас есть. Это не пустой лист бумаги. В его создании участвуют все. Но со временем я поощряю всех членов команды, чтобы они добавляли в него новые идеи и вносили поправки в уже существующие.

Эндрю: **Думаю, многие удивятся, узнав, что рок-н-ролльные альбомы и другая музыка создаются при помощи таблиц и графиков, записи целей и задач и поэтапного планирования проекта. Пожалуй, всем будет приятно узнать, что не только они ежедневно сталкиваются с этим и что такие несложные инструменты помогают создать великое произ-**

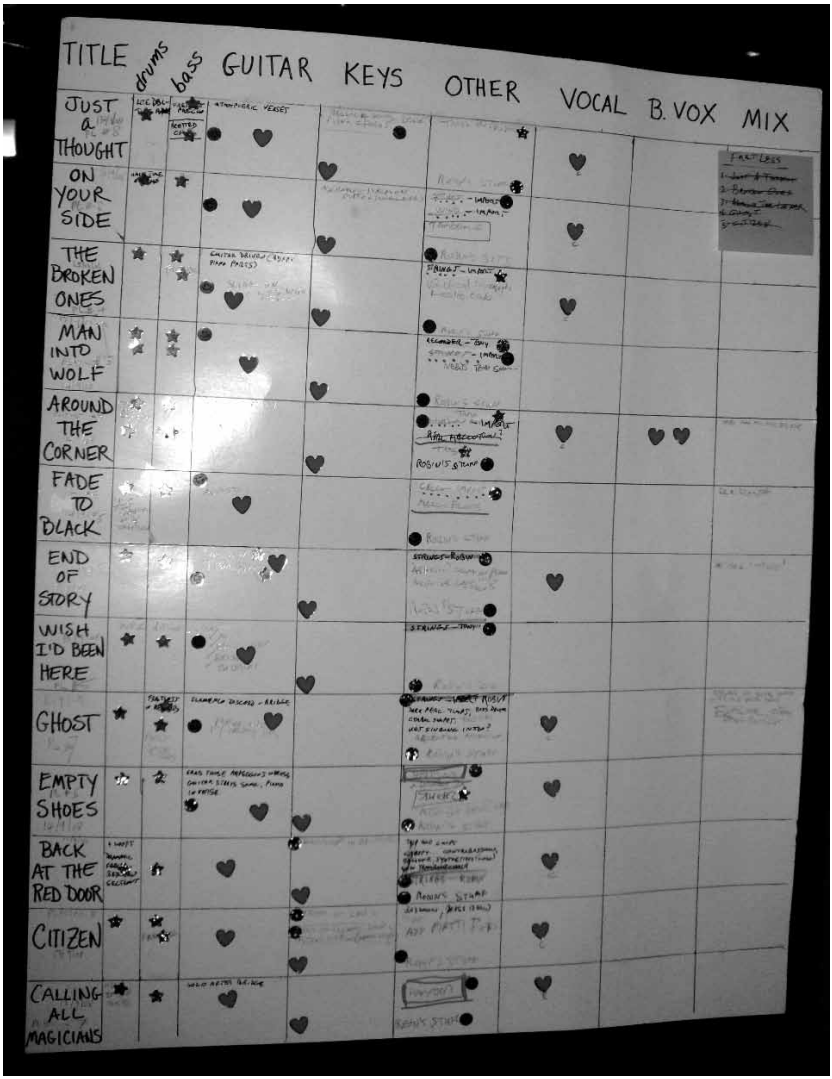


Рис. 31.1. *Подобные графики Тони использует, чтобы все в команде могли в любое время увидеть, на каком этапе находится альбом. Чем они похожи на таблицы Майка Кона? Чем отличаются?*

ведение искусства, а кроме того способствуют творческому процессу, не делая его ущербным.

Тони: Ну, любое дело, сопряженное с большими затратами (а их подразумевает и рок-н-ролл), требует организации. Был период карт-бланша, когда у менеджера Led Zeppelin стояла настоящая скульптура лебедя, полностью сделанная из кокаина (и ее даже включили в бюджет записи). Но такие послабления долго шли рука об руку с разорением музыкального бизнеса. Сегодняшний бюджет должен быть четко организованным и принятым для музыкального бизнеса, потому что нам приходится конкурировать с видеоиграми и другими мультимедийными продуктами. Поп-музыка сегодня уже не так загребает средства, как это наблюдалось пару десятилетий назад. У детей больше развлечений. Поэтому, боюсь, музыка сегодня требует эффективного продюсирования, особенно рок-н-ролл.

Но это хорошо. Люди говорят мне: «У нас достаточно денег, чтобы засадить вас и вашу команду музыкантов в студии записи на семь дней, – что вы можете нам предложить?» Я отвечаю в таких случаях честно: «Могу сделать четыре песни хорошего качества». «Но мы хотим альбом!» Я могу вести переговоры некоторое время, но если они будут настаивать на альбоме, скажем, на 12 песен, то я говорю им, что в этом случае мы сможем записать две песни на высоком уровне, а все остальное будет так себе, без всяких изысков. Дебютный альбом исполнителя Jewel («Pieces of You») был именно таким. Они потратили свой небольшой бюджет на сингл («Who Will Save Your Soul»), а все остальные треки альбома были в основном живыми записями с их клубного концерта. Вы не можете согласиться на выполнение проекта, не будучи откровенным в вопросах бюджета. Если на середине проекта у вас закончатся средства, это будет настоящей трагедией и для вас, и для вашей репутации.

Эндрю: **Как же вы всему этому научились? Ясно, что у вас не одна стратегия работы, и я знаю, что таким вещам не учат, к ним приходят самостоятельно, нередко через неудачи. Безусловно, это приобретается только с опытом. Я почти уверен, что нет ни книг, ни учебников, откуда можно почерпнуть такие знания.**

Тони: Что ж, наверное, для двадцатилетних типично хотеть стать крутым и клевым. Как раз это и не давало мне раскрыться полностью. В конце концов я понял, что потерял кучу времени и позволял плохим альбомам выходить, потому что боялся говорить правду.

Но в какой-то момент я устал быть, как бы это назвать, *трусом*, у которого не хватает смелости назвать вещи своими именами. Однажды я все-таки открыл шлюзы. Я имею в виду, что после этого уже не мог контролировать себя, и тогда мне пришлось учиться дипломатии.

Эндрю: **Но вот эти графики и таблицы – вы сами до этого дошли?**

Тони: Я придумал эти графики в 1980-е. Сегодня я вижу, что ими пользуются многие. Люди в моей команде доверяют моим личным замечаниям по поводу того, что за чем делать, а кто-то добавляет свои личные замечания к моим. Я подумал: «Почему бы нам всем не собрать наши идеи и планы в одну таблицу и не повесить все это на стену?» В первый раз, когда я решил это сделать, результат был потрясающий. Теперь, когда мне приходится работать с теми же исполнителями, они надеются увидеть график!

Но я не дохожу до фанатизма в этом вопросе. У меня есть календарь с обратным отсчетом, о котором я уже говорил. Есть и еще один, когда я работаю над штукой, которую называю «создание комбинированной песни». Например, исполнитель записал песню семь раз; у меня есть две координатные оси, на которых я отмечаю почти каждое спетое слово, чтобы потом давать перекрестные ссылки на, скажем, восемь записей. Отобрав лучшие строки, слова и слоги, я могу сконструировать «идеальную» песню. Каждый, кто хочет участвовать в создании вокальной композиции, может сделать свой график, а потом мы просто сравним наши результаты. Я стал практиковать эту «состязательную таблицу» в начале 1970-х, а теперь многие исполнители и продюсеры используют разные варианты таких таблиц.

Я считаю, что это такие интерактивные инструменты, которые помогают сплотить команду. Члены команды вовлекаются, ощущая свою причастность к процессу. И все же у меня есть право вето в таких вопросах, потому что я непосредственно отвечаю за выполнение проекта. Эти таблицы – потрясающий инструмент для работы с командой.

Эндрю: **Пожалуй, это и впрямь эффективно. Похоже, вы выработали способ наглядно представить продвижение проекта таким образом, что вся команда ощущает свою причастность. Наверное, это позволяет вам сбалансировать индивидуальность каждого члена команды. Думаю, это продойдет и для команды программистов. Когда работаешь с программистами, особенно выдающимися, бывает, что лучшие члены команды в то же время и самые «проблемные» в отношениях с коллегами. Заслужить их уважение можно, только доказав им, что тебя есть за что уважать. Знакомо это вам?**

Тони: Человеческое сознание работает порой очень странно, а творческим людям часто не хватает навыков общения; это связанно с тем, что за изобретательность и творчество отвечает довольно большая часть мозга. Такие люди не замечают, какое впечатление их бурные вспышки производят на других людей, и как они в результате этого отдаляются от команды. Они просто не видят этого. Это социальная проблема, но она напоминает ситуацию, когда имеешь дело с маленьким очень эгои-

стичным и капризным ребенком. Этим своим сравнением я не пытаюсь посмеяться над творческими людьми, но часто на эмоциональном уровне они и правда недалеки от таких детей.

Часто у них возникают проблемы с командой и, как и в случае с маленькими детьми, вам надо уверить их в том, что их любят. Ведь на самом деле, проблема сводится именно к этому. Такое поведение – это возглас: «Ты меня любишь?» И как раз ответ на этот вопрос они хотят услышать. Конечно, все мы, взрослые люди, не можем постоянно приговаривать: «Я люблю тебя, Джим». Надо показывать это как-то иначе. Сделать так, чтобы ваша высокая оценка имела довольно очевидное проявление. Например, во время собрания вы можете обращаться сначала к Джиму. Допустим, вся группа собралась у вас в кабинете. И вы говорите: «Давайте обсудим следующий этап. Я хочу услышать твое мнение, Джим». Очень вероятно, что Джиму в данный момент нечего сказать, но он обязательно отметит про себя, что к нему обратились в первую очередь.

Может быть кто-то сочтет это хвастовством, но став отцом, я научился общаться и с детьми, и со звездами. В каждой музыкальной группе, с которой мне приходилось работать, были и сознательные, взрослые люди, и парочка одержимых «звездной болезнью». Это как неофициальная иерархия, это естественно для любых форм социального взаимодействия. В успешной команде все знают, что команда не может без своей «звезды». К счастью, в большинстве случаев, такая «звезда» прекрасно понимает, что и ей не выжить без хорошей команды.

Я известный продюсер звукозаписи, но я не могу позволить себе стать «звездой» своей команды. Мне надо сохранять здравый смысл и четкость работы сознания, чтобы вести свою команду через тернистый путь к победе.

Эндрю: **Знаете, то же самое я могу сказать и про себя. В начале своей карьеры, будучи ведущим программистом небольшой команды, я вел себя как примадонна. И этим все обо мне сказано. А теперь мне не дает покоя мысль, что окружающие наверняка относились бы ко мне лучше, скажи я тогда себе: «Знаешь, что? Ты немного перебарщиваешь с требовательностью. Ты нетерпелив с людьми и чего-то требуешь от них. Но ты хорошо работаешь, поэтому они терпят тебя». Что бы вы посоветовали тем, кто узнал по этому описанию себя, что помогло бы им общаться с командой?**

Тони: Думаю, им надо научиться видеть, как их эгоистичное и раздражительное поведение влияет на команду; есть много других отличных программистов, которые с радостью будут работать в команде, после того как наш герой станет жертвой «мятежа». В этом мире, где столько

талантов, любому члену команды можно найти замену, которая будет не хуже. Поэтому каждый отвечает за то, чтобы научиться вести себя в команде, даже «звезда».

Вы постоянно конкурируете с другими первоклассными командами. Вы не можете допустить внутренних раздоров. Вы не сможете достичь общей поставленной цели, если один из членов команды постоянно подрывает командный дух, снижая продуктивность команды. Вас заменит другой, кто лучше выстраивает отношения. Любого можно заменить.

Эндрю: **Прежде чем мы закончим, хочу спросить, есть ли у вас какой-нибудь совет для хорошего музыканта, хорошего программиста или просто хорошего специалиста? Для тех, кто видит следующий этап своей карьеры в том, что делаете вы, в руководстве проектом или командой, или просто хочет убедиться, что он на верном пути? Какой совет вы бы им дали?**

Тони: Ну, сейчас много отвлекающих моментов и крайне сложно сконцентрироваться на чем-то одном. Например, мой девиз для музыкантов, которые пишут музыку на компьютере: «Овладеть компьютером гораздо легче, чем настоящим музыкальным инструментом». Смышленому восьмилетнему ребенку вполне доступна работа в Garage Band – просто перетаскивай мышью один луп за другим, да вставляй их на пустые места. Мой девиз – «Отложи мышь и возьми гитару». Нам всем угрожает утрата способности концентрироваться на чем-то одном, достигая в этом совершенства. Сегодня порой трудно побороть желание целиком отдаться катанию на скейтборде, видеоиграм или алкоголю – многие впадают в безудержное пьянство, и довольно рано. А великие музыканты во все времена очень много учились и упражнялись, стремясь к совершенству.

И я думаю, что даже если вы достигли какого-то определенного уровня (к примеру, многие юные музыканты полагают, что вполне достаточно знать 12 аккордов), не следует останавливаться на этом. Мой совет: чем больше вы знаете и чем больше учитесь, тем больше у вас шансов преуспеть в жизни. Вот и все. Поэтому посвящайте свое свободное время учебе, получению новых знаний. Я до сих пор узнаю что-то новое о своей работе, и это помогает мне добиваться еще лучших результатов.

Участники проекта

Скотт У. Амблер (Scott W. Ambler) – практический лидер Agile-разработок в IBM Software Group. Работает с пользователями IBM всего мира, помогая им улучшать работу с программами. Создал методологии Agile Modeling (AM), Agile Data (AD), Agile Unified Process (AUP) и Enterprise Unified Process (EUP). Скотт – автор и соавтор 19 книг, в том числе «Refactoring Databases»[1] (Addison-WesleyProfessional), «Agile Modeling» (Wiley), «Agile Database Techniques» (Wiley), «The Object Primer» (третье издание – издательство Кембриджского университета) и «The Enterprise Unified Process» (Prentice Hall). Скотт – старший редактор журнала «Information Week». Его личная веб-страница: *http://www.ibm.com/software/rational/leadership/leaders/#scott*. Ведет блог Agile at Scale: *www.ibm.com/developerworks/blogs/page/ambler*.

Скотт Беркун (Scott Berkun) – автор вышедших в издательстве O'Reilly бестселлеров «Making Things Happen»[2] и «The Myths of Innovation». Работал менеджером в Microsoft в 1994–2003 годах. Принимал участие в создании первых пяти версий Internet Explorer. С 2003 года – независимый писатель и лектор. Его работы появлялись в «Нью-Йорк таймс», «Вашингтон пост», журнале «Wired» и на Общественном национальном радио. Вел курс креативного мышления в Вашингтонском университете и семинары по инновациям для членов «Форчун-500», регулярно пишет для сайта Harvard Business Digital. Часто выступает в роли эксперта по менеджменту и креативному мышлению на каналах MSNBC и CNBC, а также ведет популярный блог с видеороликами, подкастами и эссе: *www.scottberkun.com/blog*.

---

[1] Скотт В. Эмблер, Прамодкумар Дж. Садаладж «Рефакторинг баз данных. Эволюционное проектирование». – М.: Вильямс, 2007.

[2] Беркун С. «Искусство управления IT-проектами», 2-е издание. – СПб.: Питер, 2010.

Барри Бём (Barry Boehm) – преподаватель программирования на факультете компьютерных наук в Университете Южной Калифорнии, а также директор Центра систем и программирования при Университете Южной Калифорнии. Доктор Бём служил в Министерстве обороны США в 1989–1992 годах, возглавляя центр информационных наук и технологий при Агентстве передовых оборонных исследовательских проектов и являясь директором отдела компьютерных технологий и программирования при управлении научно-исследовательскими и опытно-конструкторскими работами министерства обороны. В 1973–1989 годах работал в корпорации TRW, закончив работу в должности главного исследователя группы оборонных систем, а в 1959–1973 – в корпорации Rand, где стал главой отдела информационных наук. К программированию приобщился в General Dynamics в 1955 году. Его текущие научные интересы включают преобразующиеся системы и программирование в стоимостно-ориентированной среде, в том числе процессы, методы, инструменты и основополагающую теорию стоимостно-ориентированных систем и определения программ; создание, разработку, проверку и эволюцию. Его вклад в науку – это, например, семейство систем COCOMO и модели оценки программирования, спиральная модель и модель пошаговых обещаний систем и процесса программирования, а также теория W – подход к управлению системами и программами и определению требований.

Получил награду ACM за достижения в программировании и награду Харлана Миллза IEEE и степень почетного доктора компьютерных наук Массачусетского университета. Член основных профессиональных сообществ в области компьютеров (ACM), аэрокосмических технологий (AIAA), электроники (IEEE) и системотехники (INCOSE), а также член Американской национальной академии инженерных наук.

Гради Буч (Grady Booch) известен по всему миру своей инновационной работой в области архитектуры, программирования и сред совместной разработки. Посвятил всю свою жизнь улучшению искусства и науки разработки программ. Был ведущим исследователем в корпорации Rational Software с момента ее основания в 1981 году и до приобретения IBM в 2003. Сейчас трудится в Центре Томаса Дж. Уотсона при IBM в качестве ведущего исследователя в области программирования и продолжает работу над учебником по программной архитектуре, а также возглавляет некоторые программистские проекты, которые выходят за пределы немедленного вывода продукта на рынок. Продолжает общаться с потребителями, работая над реальными проблемами,

и выстраивает серьезные отношения с академиями и другими исследовательскими организациями по всему миру. Гради – один из первых авторов унифицированного языка моделирования (UML), а также один из разработчиков некоторых продуктов Rational. Был архитектором и преподавателем программной архитектуры во многих комплексных высокотехнологичных системах по всему миру и практически в любой области. Автор шести книг-бестселлеров, в том числе «The Unified Modeling Language User Guide»[1] и «Object-Oriented Analysis and Design with Applications» (обе изданы Addison-Wesley), а также постоянной колонки по компьютерной архитектуре в «IEEE Software». Живет в Торн Бридж (Колорадо) и на острове Мауи. Любит читать, путешествовать, петь, играть на кельтской арфе, занимается бихрам-йогой и ходит на каяке. Порой кажется, что законы физики к нему не применимы. Он еще жив.

Майк Кон (Mike Cohn) – основатель компании Mountain Goat Software, оказывающей услуги по консалтингу и обучению в области управления проектами. Автор книг «Agile Estimating and Planning» (Prentice Hall), «User Stories Applied: For Agile Software Development» (Addison-Wesley Professional) и «Succeeding with Agile» (Addison-Wesley). За его плечами более чем двадцатилетний опыт. Майк был менеджером по технологиям в компаниях различного уровня – от стартапов до членов «Форчун-40». Часто печатается в журналах и выступает на конференциях. Один из основателей Scrum Alliance и Agile Alliance. Ему можно написать на *mike@mountaingoatsoftware.com*.

Майкл Коллинз (Michael Collins) – ведущий исследователь в «РедДжек», консалтинговой фирме по сетевой безопасности в Мэриленде. В этой роли он разрабатывает системы мониторинга трафика и аналитические средства для очень крупных сетей. До того как начать работать в RedJack, доктор Коллинз был членом команды службы реагирования на компьютерные инциденты при Университете Карнеги-Меллона. Там он разработал инструменты и технологии для проектов DoD CENTAUR и DHS EINSTEIN. Доктор Коллинз защитил докторскую степень по электрическому и компьютерному инжинирингу в Университете Карнеги-Меллона, регулярно публикуется и читает лекции по сетевой безопасности и ситуативной информированности.

---

[1] Буч Г., Рамбо Д., Джекобсон А. «UML. Руководство пользователя». – М.: ДМК, 2001.

Марк Денович (Mark Denovich) в настоящее время живет и работает в Европе старшим бизнес-консультантом в области автосигнализации и специализируется на системах распознавания голоса. Тренирует разработчиков, занимается исследованиями и разработками. Марк начал писать программы, чтобы оплатить обучение на физическом факультете Университета Карнеги-Меллон. Ценный совет научного руководителя: «Не думаю, что вы созданы для академической науки», – и безуспешные попытки отрастить солидную бороду привели к тому, что он оставил физику и начал делать карьеру в программировании. За время обучения он усвоил, что разработка программ – отчасти наука, отчасти искусство. В реальной жизни оказалось, что это по большей части бизнес. Мечтает однажды поселиться на собственном Уолденском пруду, с быстрым Интернетом, конечно.

Билл Дипьер (Bill DiPierre) начал писать программы еще в школе более 25 лет назад. С тех пор он занимал практически все должности, существующие внутри жизненного цикла разработки программ. Живет в Филадельфии с двумя собаками, причем обе поддерживают его карьерные устремления, но заставляют помнить, что жизнь – это нечто большее, чем быть профессионалом в своей области.

Кори Доктороу (Cory Doctorow) (*http://craphound.com*) – автор фантастических романов, блогер и энтузиаст продвижения технологий. Один из редакторов популярного блога Boing Boing (*http://boingboing.net*) и постоянный автор журналов «Wired», «Popular Science», «Make», газеты «Нью-Йорк таймс» и многих других газет, журналов и вебсайтов. В свое время был директором по европейским делам Фонда электронных рубежей (*http://www.eff.org*) – некоммерческой организации, которая защищает свободу технологий в области права, политики, стандартов и соглашений. В 2007 году был руководителем фулбрайтовских программ в Анненберговском центре общественной дипломатии при Университете Южной Калифорнии.

Кори был удостоен за свои книги премий «Локус» и «Санберст», был номинирован на «Хьюго», «Небьюла» и Британскую премию фантастики. Один из основателей компании-производителя программ с открытым исходным кодом OpenCola. В данное время входит в советы и наблюдательные советы таких компаний, как Participatory Culture Foundation, MetaBrainz Foundation, Technorati, Inc., Stikkit, Органи-

зации трансформативного творчества, Areae, Анненберговский центр изучения онлайн-сообществ и Onion Networks, Inc. В 2007 году издание «Entertainment Weekly» назвало его «Уильямом Гибсоном своего поколения». Был признан в 2007 году журналом «Форбс» одной из вебзнаменитостей и одним из молодых мировых лидеров Мирового экономического форума. Сейчас он вместе с Кирби Диком работает над документальным фильмом «This Film Is Not Yet Rated» («Этот фильм пока вне рейтинга»), а с Филиппом Паррено – над фильмом «Zidane» («Зидан»), а также пишет новый молодежный роман «For the Win» («Для победы») об организации союзов в видеоиграх.

3 февраля 2008 года он стал отцом. Его дочка Поэзия Эммелин Фибоначчи Наутилус Тейлор Доктороу – настоящее чудо, затмевающее все произведения искусства и технологии.

Патрисия Энсворт (Patricia Ensworth) – президент консалтинговой фирмы Harborlight Management Services, специализирующейся на управлении рисками, глобальной логистике и мультикультурной командной работе в программировании. Автор книги «The Accidental Project Manager: Surviving the Transition from Techie to Manager» (John Wiley & Sons) и эссе в изданиях от «CIO» до «Salon» и «Natural History». Получив степень магистра культурной антропологии в Колумбийском университете, она стала работать в Merrill Lynch тестировщиком программного обеспечения. За последующую карьеру была менеджером по качеству, менеджером программ, менеджером по связям с поставщиками, тренером и консультантом. Эти позиции она занимала в Merrill Lynch,Moody's Corporation и инвестиционном банке UBS. Среди ее клиентов по консалтингу – Citigroup, Американская ассоциация менеджмента, Мемориальный центр Слоуна-Кеттеринга по лечению рака и Американский флот. Имеет сертификаты PMP, CSQE и ISTQB. Когда не общается с устройствами, оснащенными экраном и клавиатурой, часто ходит под парусом в Нью-Йоркской бухте.

Тревор Филд (Trevor Field) – британский бизнесмен с огромным опытом в области внешней рекламы. Работал и в сфере книгоиздания в Южной Африке и Великобритании. В 1990 году Тревор начал разрабатывать PlayPump (*http://www.roundabout.co.za*) – инновационную систему подачи воды, объединяющую помпу с детской каруселью. За следующие четыре года разработал и построил первые две такие помпы, которые были установлены в Масинге, сельском районе Квазулу-Наталя, провинции Южной Африки. В 1995 году решил полно-

стью сосредоточиться на PlayPump. В следующие два года объединил усилия с Полом Ристиком, основав компанию Roundabout Outdoor. В 2000 году она получила премию Рынка идей развития Мирового банка за систему PlayPump. С тех пор система неоднократно побеждала в других конкурсах по всему миру, привлекала спонсоров (с помощью неправительственной организации PlayPumps International) со всех концов света; на данный момент одно из самых значительных пожертвований (16,4 млн долларов) заявлено Лорой Буш на Экономическом форуме Клинтона в сентябре 2006 года. Сейчас Тревор работает над переносом модели PlayPump в другие африканские страны и по всему миру. На данный момент больше 1000 помп установлены в Южной Африке, Мозамбике, Свазиленде, Лесото и Замбии. В разработке эти системы в Танзании, Кении, Уганде и Эфиопии.

Карл Фогель (Karl Fogel) – уполномоченный по приложению launchpad в Canonical Ltd., автор статей о программах с открытым исходным кодом, опытный разработчик этих программ и активист движения за реформу авторских прав. Поработав с CVS, написал книгу «Open Source Development With CVS» (Coriolis; *cvsbook.com*), а затем стал ведущим разработчиком проекта Subversion в CollabNet, Inc. На основе опыта работы там написал книгу «Producing Open Source Software: How to Run a Successful Free Software Project» (O'Reilly; *producingoss.com*). Уйдя из Google, где некоторое время был экспертом по открытому исходному коду, основал некоммерческий проект QuestionCopyright.org, где продолжает работать и после поступления в Canonical в 2008 году. Регулярно выступает в печати и с лекциями по реформе авторских прав и по применению принципов открытого исходного кода не только в программировании.

Питер Глюк (Peter Glück) – старший инженер-программист Лаборатории реактивного движения НАСА в Пасадене (Калифорния). Питер возглавлял весь программистский отдел в проекте посадочного модуля «Феникс» и за последние 20 лет участвовал во многих межпланетных миссиях НАСА. По контракту он работает с многими программистскими и технологическими компаниями, осуществляя разработку и внедрение с 2002 года. Лауреат индивидуальных и групповых премий от таких аэрокосмических организаций, как НАСА, АИАиА и ЛРД, часто выступает в печати и с лекциями по вопросам программирования. Бакалавр математики Калифорнийского университета в Нор-

тридже и магистр аэрокосмического приборостроения Университета Южной Калифорнии.

Джеймс У. Греннинг (James W. Grenning) – основатель агентства Renaissance Software Consulting, где работает тренером, коучем и консультантом по всему миру. Обладая более чем 30-летним опытом работы в программировании – как в разработке, так и в руководстве, Джеймс привносит в команды программистов и управление ими все богатство знаний, умений и креативности. Цель Джеймса – обучать команды разработчиков улучшенным методам программирования и управления проектами. Его профессиональный интерес начался с встроенного ПО, и теперь он стремится привнести в этот многообещающий мир методы Agile-разработки. В данное время пишет книгу о применении управляемой тестами разработки к встроенному ПО. Джеймс – один из первых тренеров и коучей по экстремальному программированию, всегда работает на пределе. Он изобрел Planning Poker – Agile-технику оценки, применяемую во всем мире. Также он участвовал в создании Манифеста Agile-разработки ПО (*http://agilemanifesto.org*).

Марк Хили (Mark Healey) – креативный директор Media Molecule. Придя в индустрию игр в 1988 году, оказался одним из самых талантливых разработчиков игр. Художник по образованию, он и программировал, и разрабатывал дизайн, и писал аудиотреки к играм. В 1994 году поступил в Bullfrog Productions и работал над такими их хитами, как Theme Park, Magic Carpet и Dungeonkeeper. В 1997 году из Bullfrogперешел в Lionhead Studios, где работал над Black & White и Fable. Участие в работе над культовым интернет-симулятором Live for Speed привило Марку вкус к разработке в более мелких компаниях. В 2002 году он приступил к работе над собственной игрой Rag Doll Kung Fu, создав графику, код, дизайн и музыку лишь с помощью нескольких друзей. Сочетание играбельности и своеобразного юмора оказалось победоносной формулой для игры, и она была номинирована на премию «Develop Awards Best New PC IP and Innovation». Разработка и последовавший за ней успех Rag Doll King Fu показали Марку, что ему лучше всего работается в небольшой группе, трудящейся интенсивно, но весело. Вдохновившись, он создал новую студию разработки Media Molecule – вместе с друзьями, приложившими руку к успеху Rag Doll Kung Fu: Алексом Эвансом (Alex Evans), Дэвидом Смитом (David Smith) и Каримом Эттуни (Kareem Ettouney), а также Крисом Ли (Chris Lee). Первая игра студии – LittleBigPlanet – вышла в ноябре 2008 года.

Ауке Джилдерда (Auke Jilderda) – специалист по продажам и менеджер по технической поддержке в CollabNet. Свою карьеру Ауке начинал в Philips Research, изучая инжиниринг ПО с открытым исходным кодом, чтобы понять, почему некоторые проекты имеют оглушительный успех, а каким-то не хватает импульса. Определив ключевые моменты, которые делают проекты с открытым исходным кодом успешными, он перешел в Philips Medical Systems, чтобы на практике применить эти ключевые моменты к «внутреннему коду» в крупной компании, чтобы улучшить ее способность к взаимодействию через организационные и пространственные границы. Попутно работал над стратегией разделения в программировании, выясняя, где, когда и для чего можно внедрять в продукт решения с открытым исходным кодом как жизнеспособную альтернативу собственной разработке или покупке у третьей стороны.

Четыре года назад Ауке поступил в европейский отдел CollabNet и теперь занимается как инжинирингом продаж, так и стратегическим техническим аккаунт-менеджментом и консалтингом по инжинирингу совместного программирования.

Энди Лестер (Andy Lester) – разработчик ПО и писатель. Более 20 лет работал с большими и малыми командами, в том числе руководил ими, от небольших офисов до всемирного сотрудничества. Энди ведет блог о технологиях The Working Geek (*http://theworkinggeek.com*). Его опыт в области найма на работу отражен в книге «Land the Tech Job You Love» (Pragmatic Bookshelf).

Алекс Мартелли (Alex Martelli) – член фонда Python Software. Работает в Маунтин-Вью (Калифорния), в штате Google, чаще всего как младший технический руководитель, но в данный момент является старшим инженером по персоналу. Алекс написал книгу «Python in a Nutshell» (O'Reilly), а также много других книг, статей, эссе, выступал на конференциях по Python, численным методам и техническому управлению разработкой программ. Алекс – инженер-электрик, магистр Болонского университета; до Google он работал в Texas Instruments, IBM Research, think3, а также как независимый консультант. Преподавал программирование, методы разработки, объектно-ориентированное проектирование и числовые методы в Феррарском университете и других учебных заведениях.

Стив Макконнелл (Steve McConnell) – генеральный директор и ведущий инженер-программист Construx Software. Пишет книги и статьи, консультирует, ведет группы и осуществляет общее управление программированием в Construx. Автор книг «Software Estimation: Demystifying the Black Art»[1], «Code Complete»[2], «Rapid Development», «Software Project Survival Guide»[3] (Microsoft Press) и «Professional Software Development»[4] (Addison-Wesley Professional). Первые две книги получили премию «Jolt Excellence» журнала «Software Development» как лучшие книги года по программированию.

Стив в индустрии программ для персональных компьютеров с 1984 года и за это время занимался оценкой проектов, методами создания программ, скоростными и Agile-методологиями разработки, руководил аутсорсингом ПО. В 1998 г. читатели «Software Development» назвали Стива одним из самых влиятельных людей в индустрии ПО наряду с Биллом Гейтсом и Линусом Торвальдсом. В 1998–2002 годах Стив был главным редактором журнала «IEEE Software». Является одним из членов экспертного совета проекта SWEBOK, возглавлял Комитет профессиональных методов компьютерного сообщества IEEE. Стив – бакалавр Уитманколледжа и магистр программирования Сиэтлского университета.

Энди Орам (Andy Oram), редактор O'Reilly Media, автор статей о технологиях и их общественном применении с середины 1990-х гг. (*http://www.praxagora.com/andyo/professional/article.html*). 10 лет был техническим писателем в компьютерной индустрии, а затем вступил на книгоиздательское поле, чтобы достичь тех же целей: дать возможность программистам и пользователям компьютера эффективнее выполнять свои задачи. В O'Reilly Энди много лет специализировался в области ПО с открытым исходным кодом. Был редактором первых в Америке книг о Linux и таких основополагающих работ, как «Peer-to-Peer», «Mastering Regular Expressions»[5], «Intellectual Property and

---

[1] Макконнелл С. «Сколько стоит программный проект». – СПб.: Питер, Русская редакция, 2007.

[2] Макконнелл С. «Совершенный код». – СПб.: Питер, 2007.

[3] Макконнелл С. «Остаться в живых! Руководство для менеджера программных проектов». – СПб.: Питер, 2006.

[4] Макконнелл С. «Профессиональная разработка программного обеспечения». – СПб.: Символ-Плюс, 2007.

[5] Фридл Дж. «Регулярные выражения». – СПб.: Символ-Плюс, 2008.

Open Source» и «Beautiful Code»[1]. Также интересуется образованием в онлайн-сообществах, что представлено на *http://www.praxagora.com/community_documentation/*. Постоянный член организации Computer Professionals for Social Responsibility (CPSR, Компьютерные профессионалы – за ответственность перед обществом).

Тим О'Рейли (Tim O'Reilly) – основатель и генеральный директор O'Reilly Media, лучшего в мире компьютерного издательства, по мнению многих. Компания также публикует книги в Сети посредством O'Reilly Network и организует конференции по технологическим вопросам. Тим – активист движения за ПО с открытым исходным кодом, открытыми стандартами, а также за разумные авторские права.

С 1978 года Тим ведет компанию к ее главной цели: стать катализатором технологических изменений путем сбора и передачи знаний «альфагиков» и других новаторов. Его активные связи с технологическими сообществами касаются как разработки продукта компании, так и маркетинга. Тим создал культуру, ключевые принципы которой – пропаганда, создание мемов и евангелизм.

Тим входит в число доверенных лиц как Общества Интернета (Internet Society, ISOC), так и Фонда электронных рубежей (Electronic Frontier Foundation) – двух организаций, которые следят за тем, чтобы Интернет выполнял возложенные на него задачи. Работал в Macromedia до слияния компании с Adobe. Сейчас сотрудничает с CollabNet.

Тим с отличием окончил Гарвардский колледж в 1975 году, он бакалавр классической филологии. Архив онлайн-статей Тима, его речей и интервью можно найти на его архивной странице (*http://tim.oreilly.com*).

Доктор Мария Э. Пенедо (Dr. Maria H. Penedo) – Выдающийся технический сотрудник и старший исследователь корпорации Northrop Grumman (NG). В NG была ведущим специалистом по проектированию, внедрению и использованию инжиниринга ПО и информационных технологий в программах, проектах и ИТ-организации, особенно в процессо-ориентированных средах, архитектуре, интеграции, совместной работе, агентах и технологиях Web 2.0. Свой энтузиазм и страсть она направляет в русло новых технологий и тенденций в мире науки и программирования. Ее работы в области распределения совместной работы

[1] Орам Э., Уилсон Г. «Идеальный код». – СПб.: Питер, 2009.

были в 1998 году отмечены премией за инновации Председателя NG; среди прочих наград – многочисленные премии за внутренние исследования и разработки; премия Женщине года в области технологий, награда за преподавательскую деятельность. Признанный лидер в международном сообществе науки о программировании, активно участвует в работе большинства крупнейших обществ этой отрасли. Пишет и читает лекции по всему миру, работает в программных комитетах крупнейших национальных и международных конференций. Была редактором журналов «IEEE Transactions on Software Engineering» и «Wiley Software Process: Improvement and Practice Journal», а также Почетным членом-корреспондентом Компьютерного общества IEEE. Бакалавр математики, магистр компьютерных наук (степень получена в Бразилии) и доктор компьютерных наук в Университете Лос-Анджелеса.

Доктор Карл Ремер (Dr. Karl Rehmer) защитив докторскую степень по математике, несколько лет преподавал математику и компьютерные науки. Оставив в 1988 году академический мир, чтобы получить работу в таком месте, где его жена могла бы работать над своей докторской диссертацией по компьютерам, поступил в Honeywell Air Transport Systems в Финиксе (Аризона). Работая там, входил в команду HADS, которая разрабатывала инструменты для разработки авиационного программного обеспечения для Боинга 777. Затем доктор Ремер перешел в исследовательский центр Honeywell, где разработал планировщик для встроенной операционной системы реального времени Deos, используемой многими авиационными приложениями. В 1996 году устроился в компанию DDC-I, производящую ПО в области безопасности. Является главным архитектором проекта DDC-I Mixed Language Debugger. Помимо этого, работает по ряду субконтрактов с клиентами DDC-I. Сейчас добавляет поддержку новых языков и операционных систем в DDC-I Mixed Language Debugger.

Эрик Ренки (Eric Renkey) – программист-разработчик, в данное время управляет небольшой командой в интегрированной энергетической компании. Программировать научился много лет назад на семейном Atari 800, профессионально занимается ИТ с 1995 года. Работать с ним – одно удовольствие, и он охотно рассказывает о своем опыте. Бакалавр психологии Пенсильванского государственного университета.

Нед Робинсон (Ned Robinson) окончил Хантер-колледж в Нью-Йорке, став бакалавром английской литературы. После компьютерной революции 1980-х решил изменить карьерные приоритеты и самостоятельно научился программировать. Сначала работал с Lotus 123 и dBase 3, но быстро прогрессировал, перейдя на C и C++. До начала 1990-х работал с несколькими банками на Уолл-стрит. Затем стал ведущим разработчиком и менеджером продукта в Sheridan Software – независимом поставщике ПО для Microsoft, который специализировался на стоимостно-ориентированных решениях VBX и DirectX.

Пианист с классическим образованием, Нед всегда был заинтересован в объединении своих технических навыков и любви к музыке. В 1995 году он основал RePlay Technologies – программистскую фирму, ориентированную на практического музыканта. RePlay Technologies работала пять лет, затем Нед продал ее и вернулся к работе в финансовом секторе. В одной небольшой компании, производившей программы для финансистов, он встретил Эндрю Стеллмана и, как и Эндрю, стал горячим сторонником программного процесса.

Разработчик в душе, он рассматривает процесс как способ вернуться в сообщество разработчиков, поскольку хороший программный процесс позволяет разработчикам блеснуть, используя свою невероятную изобретательность для претворения в жизнь бизнес-требований.

Джоанна Ротман (Johanna Rothman) помогает лидерам решать проблемы и использовать возможности. Консультирует, читает лекции и пишет об управлении высокотехнологичной разработкой продукта. Благодаря ей менеджеры, команды и организации становятся более эффективными, применяя ее практический подход к вопросам проект-менеджмента, управления рисками и персоналом. Джоанна выпускает ежемесячную рассылку и подкаст «The Pragmatic Manager», ведет два блога: Managing Product Development (управление разработкой продукта) и Hiring Technical People (прием на работу технического персонала). Написала несколько книг: «Manage Your Project Portfolio: Increase Your Capacity and Finish Projects» (Pragmatic Bookshelf), «Manage It! Your Guide to Modern, Pragmatic Project Management» (Pragmatic Bookshelf) – эта книга получила премию «Jolt Productivity Award» в 2008 году, в соавторстве с Эстер Дерби (Esther Derby) – «Behind Closed Doors: Secrets of Great Management» (Pragmatic Bookshelf), «Hiring the Best Knowledge Workers, Techies & Nerds: The Secrets and Science of Hiring Technical People» (Dorset House) и «Corrective Action for the Software Industry» (Paton Press) – в соавторстве с Дениз Робитайл (Denise Robitaille).

Нил Дж. Сигель (Neil G. Siegel) – вице-президент и главный инженер ИТ-отдела корпорации Northrop Grumman (NG). В его активе многие успешно внедренные системы для армии и разведки, в том числе армейская система Blue-Force Tracking и беспилотное воздушное судно Hunter. Программы, которые он создавал, выиграли множество наград, в том числе «Crosstalk Award» в 2001 году как один из пяти лучших программных проектов при правительстве США и «Monticello Award» в 2003 году (за информационную систему, которая оказывает прямое и непосредственное воздействие на человеческие жизни). Его изобретения позволили компании заработать миллиарды долларов, а ему принесли более дюжины патентов.

Помимо всего прочего, в 2005 году был избран в Национальную академию инженерных наук США.

Том Тарка (Tom Tarka) работал в MP3.com всего пару лет, но этот опыт – люди, место, работа – оставил неизгладимый след на его личности. После ухода он собрал пожитки и большую часть года провел, разъезжая по Штатам в «фольксвагене кемпере» 1971 года выпуска, пока наконец не обосновался на восточном побережье. Поработав механиком, позднее получил шанс, смахнув пыль со своего диплома химика, устроиться на работу в интересовавшей его с детских лет области – энергетике. В настоящее время работает над источниками энергии нового поколения и глобальным изменением климата в американском Министерстве энергетики. Лицензированный профессиональный инженер.

Том живет в Питтсбурге, у него там дом и небольшая коллекция микроавтобусов «фольксваген».

Тони Висконти (TonyVisconti) – музыкальный продюсер с сорокалетним стажем. Продюсер 12 альбомов Дэвида Боуи, 10 альбомов T. Rex и многих других британских исполнителей. Последние два года Тони работал с Моррисси Razorlight, Алехандро Эсковедо, Anti-Flag и Fall Out Boy. В издательстве Harper Collins вышла его автобиография – «Bowie, Bolan and the Brooklyn Boy».

Карл Вигерс (Karl Wiegers) – главный консультант образовательной и консалтинговой компании по программированию Process Impact (Портленд, штат Орегон) по спецификациям, оптимизации процесса и проект-менеджменту. Автор книг «Software Requirements»[1], «More About Software Requirements» (обе изданы Microsoft Press), «Peer Reviews in Software» (Addison-Wesley), «Practical Project Initiation» (Microsoft Press) и «Creating a Software Engineering Culture» (Dorset House), а также ряда статей по многим аспектам программирования, химии и военной истории. Доктор наук в области органической химии, что считает отличной стартовой площадкой для карьеры в программировании. Его сайт: *http://www.processimpact.com*.

[1] Вигерс К. «Разработка требований к программному обеспечению». – М.: Русская редакция, 2004.

Алфавитный указатель

Цифры

11 сентября 2001 года, атака террористов, 430, 436
80/20, принцип, 88

A

AFATDS, информационно-управляющая система, 355
Agile-разработка
  (*см. также* экстремальное программирование), 242
  беспорядочные пары, 447
  методы, 200
  планировние, 194, 199
  проблемы, 447
  простота, 341
  проекты, 158, 183
  и плохо работающие члены команды, 335
  качество, 338
  контроль расписания проекта, 339
  уклонение от общих мест, 341
  универсальность, 199
AIMS, система управления, 381
Ajax, 188
Apache, 34, 35, 124, 131, 145
Apple, 168

B

Betamax, видеомагнитофон, 121
BTK-версии, 480, 486

C

CISC (Complex Instruction Set Computing) процессоры, 544
CollabNet, 146
Constructive Cost Estimation Model, (COCOMO), 280
Contribulyzer, инструмент, 347
  запись действий, 353
  недостатки, 354
Creative Commons, проект, 205
CMM, 524

D

D.A.M.-диски, 119
DDC-I, поставщик компиляторов, 380
DOS, ОС, 69

F

Firefox, 145
Framalang, вики, 364

G

Global Network Navigator, сайт, 36
Google, 168
  команды разработки, 344
GPL, лицензия, 29
groupware, лицензия, 446

H

Honeywell Air Transport Systems, авиатранспортные системы (*см. также* команда HADS), 380

I

IBM, 146, 164, 168
Inner Source, проекты, 145, 156
  влияние на разработку ПО, 155
  документация, 154
  доступ к исходному коду, 151
  инфраструктуры сотрудничества, 153
  ключевые аспекты успешности проектов, 152
  коммуникации и рассылки, 151
  открытость разработки, 148, 154
  политические аспекты, 150
  применение, 153
Internet Engineering Task Force (IETF), комиссия, 35
Internet Explorer, 49, 343
iPod, 32

J

JavaScript, 130, 188

K

Kindle, электронная книга, 32
Kmail, проект, 146

L

Line Replaceable Unit (LRU), 382
Linux, 145
LittleBigPlanet, компьютерная игра, 53

M

McCabe, инструменты, 465
MAKE, журнал, 39
Media Molecule, 57
  (*см. также* игры, создание)
Media Molecule, компания, 53
Microsoft, 35, 49, 67, 127, 146
Motif, библиотека, 504
MP3.com, компания, 115
  (*см. также* команда MP3.com)
MP3-плееры, 122
MySpace, 57

N

Netscape, 35, 49
Northrop Grumann, оборонно-
  техническая компания, 211, 222
Novell, 96

O

Office, проект, 96, 186
Open Source, 28, 31
  методы распределенных команд, 156

P

Perl, язык, 33, 116, 131
Play Station, 53
PowerPoint, 59
push-технология, 50

R

Rag Doll Kung Fu, игра, 54
Rational Unified Process (RUP), 344, 520
RISC-процессоры, 543, 555
RSS-каналы, 50
Ruby on Rails, 171

S

scrum-методы, 196
Slashdot.com, 207
Sony, компания, 58, 121
Street Fighter, игра, 54
Subversion, инструмент, 347
  коммит из проекта, 356

T

TRW, проект, 278

U

UNIX, 29
  внедрение, 286
  возможности, 282
  переоснащение, 542
  принятие пользователями, 285
User Configurable Code (UCC), 386, 384

V

Visual Basic, 67
Vivendi Universal, 141
VMS, ОС, 285

W

Web 2.0, 39
wiki, движок, 29
Word, 96

X

Xerox, 168
XML
  мастер-файлы для книги, 361
  проблемы с работой в, 36

Y

Yahoo!, сайт, 36
YouTube, 57, 204

A

абстрагирование, 169
автоматизация, 283
автоматическое тестирование, 193
авторское право в Интернете, 203, 210
адаптированность социальная, 87
активность широких масс, онлайн, 553
анализ
  затрат и выбор технологий, 284
  основных причин ошибок, 462
аппаратные испытательные стенды, 299
архитектура систем, 35, 166
  как средство управления, 172
аутсорсинг, 162

Б

базовый документ, 328
Балмер, Стив, 35
Безумный Ляпник (БЛ), 480
бейсбольные агенты, 46
библиотека Motif, 543
бизнес-план, 57
благотворительная организация, управление, 230
благотворительность, 223
блоги;вики, 345
бортовое ПО для Боинга-777, 379
борьба за власть между членами команды, 67

В

ваби-саби, 48, 52
  команда Internet Explorer 4.0, 49, 52
варианты разработки, 246
  ограничения требований, 321
  в компании со слабо проработанными процедурами, 428
вдохновление команды, 262
взаимодействие
  личное общение в команде, 269
  облегчение, 262
  преимущества и ограничения электронной почты, 287
видение личное, 449
вики-движок, 364
Википедия, 29, 205
виртуальные команды, 270
власть против открытости и ответственности, 273
влияние проекта на рост занятых в нем сотрудников, 539
внедрение новых технологий, 285
  вовлечение пользователей, 286
  понимание интересов пользователей, 287
  процесс принятия решений, 263
вознаграждение, 94
ВОИС (Всемирная организация интеллектуальной собственности), 204, 210
воодушевление людей, 95
вооруженные силы, работа с, 217
выборочная мотивация, 47
выделиться желание, 86
высокопродуктивные команды
  защита и сохранение, 264
  определение, 258
  процент команд, определяемых как, 261
  создание, 261
выстраивание общих целей, 212
вычитки, 326
выявление аномалий, 372

Г

генерация идей
  команда создателей игр, 59
географически удаленные команды, 270
герои, реальная жизнь и кино, 47
гибкая среда разработки
  среда команды HADS, 396
гибкость решений, 199
гики, 86
  этикет, 88
гиперпродуктивные организации, 172
групповое мышление, 451
группы разработчиков, 59

Д

Давидов, Уильям, 36
дедлайн, 174
  и забота о качестве, 188
  стресс, 56
  строгость, 301
деловое правило, 322
демоверсия, 58, 177
децентрализованные проекты, 343
джаз, 452
Дженин, Гарольд, 28
Джобс, Стив, 37
динамика команды, 84
дипломатия, роль в руководстве команды, 296
диски D.A.M. (цифровая автоматическая музыка), 119
дифф, 356
  сравнение версий, 331
добрая воля, 262
доверие в распределенных командах, 163
документация
  архитектура систем и, 166
  в экстремальном программировании (XP), 248
    легковесный подход, 248
    проектирование, 244
  уровни, в командах разработчиков, 154
доска задач, 196
дух командный, 176

Ж

желание выделиться, 86
«железный закон» поддержки ПО, 289

З

задачи, разделение на небольшие порции, 466
зануды, 128
заслуги профессиональные, 161
защита команд, 169
защита от ошибок, 298
Зюст Ван, Сандер, 117

И

игры, создание, 47, 59
  Media Molecule, команда, 57
  Rag Doll Kung Fu, проект, 55
  генерация идей, 59
  дедлайны и стресс, 56
  качество участников команды, 62
  контент, созданный пользователями, 57
  лидерство в команде, 65
  уважение к членам команды, 63
  управление концепцией, 60
  эволюционирующее проектирование, 60
    контроль качества, 61
идеальные команды, построение, 81
  борьба за власть между членами команды, 67, 69
  подготовка членов команды, 79
  подпитка эго, 79
  политика, 70
  работа с трудными членами коллектива, 73, 77
  солидарность, 77
идеальные условия разработки, 167
идеи, 59, 100
идентичность командная, 242
иерархия руководства и экспертных знаний, 274
изменения, как препятствия, 411
индивидуальный стиль работы, 165
инновации, 100
  неудачи, 170
инструменты
  McCabe, 465
  и команды, 321
  совместной работы, 345, 369
    Contribulyzer, 355
    перевода, 341
    коммит-письма, 355, 360
    принципы, 366, 369
    проекты с открытым исходным кодом, 346
    трансляционные интерфейсы, 361
  сотрудничества, 153
интеграция, 193
  программы НАСА, 304
интеллект, 34
интеллектуальная собственность в Интернете, 203, 210
интересы общие, 93
Интернет и авторское право, 203, 210
интерфейсы перевода
  инструменты, 363, 366
  проблемы, 363
  техническая структурная база, 361
инфраструктура, 153
искусство руководителя, 28
испытания, 49
исследовательские команды, 378
  защита, 374
  конфликты среди членов, 376
  модульность системной архитектуры, 376
  обнаружение вторжения, 372
  проблема требований, 375
  роли членов команды, 375
  умения членов команды, 374
исследовательские программы, 373
исходный код, открытие доступа, 151
итеративная разработка, 243
  начало с первой итерацией, 246
итерации, 60

К

каналы, 50
каскадная модель разработки ПО на основе требований, 278
качество
  и команда Rock Ridge, 487
  и командный дух, 189
  и методология agile-разработки, 338
  и проекты ЛРД НАСА, 300
  как соответствие требованиям, 258
  методы, разница в качестве, 192
  отслеживание в проектах, 61
  проверки кодов, 462
  проекты для военных, 217
  против скорости, 457, 468, 470, 475
  процедуры обеспечения качества в Rock Ridge, 506

соотношение, 188
тестирование ПО, 511, 537
классы пользователей, 314
код, формируемый пользователем, 384
коллективная деятельность, ограничения, 203
коллективная работа с компьютером (Computer Supported Collaborative Work (CSCW)), 446
команда проекта HADS, 379, 397
завершение, 397
исправление ситуации с проектом, 385, 388
первый состав команды, 383, 385
предыстория, 380, 383
работа с пользователями, 388, 393
факторы, формирующие идеальную команду, 394, 397
команда MP3.com, 113, 143
дружба, 143
запуск сервиса, 131
избранные против зануд, 128
иски по нарушению авторских прав, 137
риск исков, 124–125
урегулирование исков, 141
конвертирование компакт-дисков, 127
конфликты среди сотрудников, 125
новая рабочая среда, 134
новый проект, новая команда, 117, 124
общение с соседями по офису, 135
охлаждение к работе, 140
переоценка ценностей, 139
перестановки в руководстве, 129
повышение в должности, 133
прием на работу, 116
продажа бренда, 141
производительность труда, 134
реакция руководства на ошибки, 132
сокращения и трения, 138
команда компании Rock Ridge, 477, 510
знакомство, 479
конец работы, 508
процессы, 484, 489
социальное взаимодействие, 494, 489
команда проекта Candygram, 480
командная идентичность, 262
командная культура, 162, 236
командный дух, 190
Agile-разработка, 338
бюджет компании, 275
и качество проекта, 189
мотивация и, 266
команды
базирующиеся в разных местах, руководство, 296
идеальные, 67
качество участников, 62
обобщение мнения, 539, 556
оценка квалификации потенциальных участников, 63
партнерство с другими компаниями, 212, 217
привлечение правильных людей в, 63
проблемные участники в команде проекта ППР, 286
Проект повышения продуктивности разработки ПО (ППР), 283
против групп, 459
работа в очень крупной компании, 221
размер, 60
создание, 91
состав, 251
трения, 150
уважение к членам, 63
характеристики, 168
коммерческое применение инноваций, 289
коммит-письма
проверка коммитов, 359
проект GNU Emacs, 357
сообщение с диффом, 358
рассылка, 355
коммуникабельность, 85
коммуникации
и новый стиль руководства, 553
устранение барьеров, 154
компромиссы, 165
комфортность, 85, 106
конструктивно-сменный узел (КСУ), 381
контакты с правительством, 227
контент, создаваемый пользователями, 57
Kung Fu Rag Doll, игра, 58
конференции внутренние, 153
конфликты в командах, 46, 75, 89, 125, 159
источники, 213
обсуждение на собеседованиях, 92
реакция, 84
концентрация на работе, 169

концепция, 60
в роли средства управления проектом, 449
управление в командах, 60, 62
красота, применительно к командам, 48
критика, 87
критический путь, 198
круговая игра, 497
культура командная, 172, 236, 251
MP3.com, 126, 143
изменения в ходе проекта, 283
команды, где работают в удовольствие, 168
конфликты, 279
концентрация на работе, 169
крупные команды, 165
наследие, 171
отношение к неудачам, 170
попытка изменить, 278
родовая память и архитектура систем, 166
социальные отношения в командах, 164
условия, обеспечивающие гладкость процесса разработки, 167
культура поведения, 90

Л

Лао Цзы, 32
ЛВС
внедрение, 283
выбор, 284
лидерство, 101, 296
и процесс, 297
коллективное, 34
команды игр, 65
личные отношения и, 169
помощь плохо работающим командам, 274
управление компанией, 28
Тим О'Рейли, 39
лифтовый ответ, 195
лицензии в Интернете, 205
личностные качества членов команды, 295
несовместимость, 445
продюсерская команда, 575
лишняя работа и «позолота», 340
лучшие методы
знакомство участников команды, 260
формирование командной идентичности, 263
цели команды и руководства, 260

М

Маслоу, пирамида потребностей, 84
мебель офисная, 176
методы оценки эффективности, 174
миссия, пропаганда, 214
модель водопада в разработке программ, 297
модель собственности и контроля, децентрализованная, 153
модульные тесты, 304
мотивация, 109, 215
возвышенные цели, 184
командный дух, 266
общей для команд и отдельных людей, 213
пропаганда миссии вашей компании, 215
работа во имя высшей цели, 216
музыка, продюсирование, 577
важность больших познаний, 577
влияние неэффективного общения на команду, 564
если проект идет не в том направлении, 566
завоевание авторитета в команде, 570
качества руководителя проекта, 563
команда довольна, но результат не удался, 567
компромисс между желаниями группы и фирмы звукозаписи, 568
межличностные проблемы в команде, 575
мужество высказать собственное мнение, 574
расхождение во мнениях в команде, 571
уважение к знаниям, 564
что делают продюсеры, 562

Н

наблюдательность, 104
НАСА, команды Лаборатории реактивного движения (ЛРД), 293, 309
дедлайны и стресс, 301
защита от ошибок и тестирование, 298, 299
идеальное сочетание талантов и личностных качеств, 295
качество и тестирование, 304, 308
конфликты и различия взглядов, 295

лидерство, 296
повторно используемое ПО, усилия на разработку, 301, 303
проблемы формирования команд, 295
проверка качества, 300
различные цели у членов команды, 300
характеристики лучших команд, 309
научная обработка данных, 543
небольшие партии (в XP), 243
невнимание, 85
недовольство сотрудников, 96, 126
недоразумения, 188
непрерывная интеграция, 192
и программы ЛРД НАСА, 304
нестандартные решения, 100
нетерпимость, 61
неудачи
боязнь, 170
в проектах ПО с открытым исходным кодом, 146
Ницше, Фридрих, 45
нормы общественные, 85
НПО (неправительственные организации), 206
нулевая итерация, 247

О

обнаружение вторжения, 346
оборонные технологии, ограничения ПМТО, 297
обработка исключений, 383
обратная связь, 153
обслуживание ПО, 289
«железный закон», 266
общение
влияние на команду, 564
в открытой рабочей среде 252
облегчение в командах экстремального программирования (XP), 243
программирование в парах, 249
членов команды, 84
общественные нормы, 85
общие интересы, 93
общие места в Agile-разработке, 341
окупаемость проекта, 186
оперативные страницы, документация, 545
операционные издержки, 203
Организация Объединенных Наций, 204
О'Рейли, Тим, 27
открытая рабочая среда (в XP), 243, 252
открытость, 554
и ответственность за действия, 272, 274
открытый исходный код, 28
в корпоративных проектах, 145
офисная мебель, 176
оценки, 47
возможностей, 173
результатов, 177
таланта, 46
ошибки, 110, 158

П

парное программирование, 269, 447
новички и эксперты учатся друг у друга, 454
трудности, 455
патчи, 347, 356
перестановки в руководстве, 129
перечни функциональности (в XP), 242, 246
пирамида потребностей Маслоу, 84
планирование, 201
выбор стандарта качества, 187
в начале проекта, 192
выяснение целей, 188
доски задач, 196, 199
и Agile-разработки, 194, 200
продюсерская команда, 572
разница в качестве кода, 192
точность планирования, 199
улучшение, 195
цели, 184, 186
планирование (в XP), 243
повторное использование, 155
и разработка ПО в ЛРД НАСА, 303
повышение в должности, 133
повышения продуктивности разработки, 283
вовлечение пользователей и внедрение новой технологии, 286
воспитание начальника, 283
инновации, сроки и коммерческое применение, 289
ключевые факторы создания идеальных команд, 291
обработка текста, 288
отдельные кабинеты, 284
сторонники, 290
требование применения разрабатываемых инструментов, 287
технологии
внедрение новых, 285

проблемы быстрой смены, 290
электронная почта и продуктивность, 287
подготовка членов команды, 79
поддержка ПО
«железный закон», 289
проекты с открытым исходным кодом, 346
«позолота», 340
поиск виновного, 110
политика офисная, 70
защита команд, 169, 170
принуждение и негативная реакция, 150
помощь, 86
понижение в должности, 134
постановка цели, 98
Постел, Джон, 35
постепенный процесс, 248
потенциал, осознание, 98
потребитель как член команды, 242
требования, 250
потребности социальные, 84
Поуп, Александр, 31
почтовые рассылки в проектах разработки, 151
поэтапный выпуск, 405
Правила международной торговли оружием (ПМТО), 297
право вето, 93
практики разработки ПО
для гиперпродуктивных организаций, 172
слишком жесткие или слишком мягкие подходы, 165
предвидение, 120
предметные знания
ограничение рамками экспертных областей, 264
получение, 218
разный язык пользователей и инженеров, 221
стирание барьеров между техническим знанием и, 219
презентации, 59
препятствия на пути к изменениям, преодоление, 443, 456
Agile-разработка, 447
методики, применение, 447
обучение в командах, 454
парное программирование, 455
прекращение бесперспективных проектов, 449
примеры великих команд, 453
проблемы человеческого фактора в командах, 446, 453
сложности в управлении, 452
привлечение специалистов, 63
прием на работу, 91, 94, 143
критерии, 47
признание компетентности, 80
принцип общественного управления в свободном ПО, 148
приспосабливаемость, 86
«проблема курилки», 163
проблемы, 174, 177
с кодом, 249
провалы, 35, 173, 181
личная вина, 180
оценка возможности, 173
ошибки и отношение к ним, 109, 132
причины, 173, 178
проверки, 241
коммиты в проектах с открытым исходным кодом, 359
проверки кода
в проектах НАСА, 306
в проектах с открытым исходным кодом, 359
в стиле Фагана, 464
обязательные, в Google, 340
и экстремальное программирование, 241, 245
требований до начала программирования, 323
программа обучения работе в команде, 267
программирование в парах
замена проверкам кода, 245
прорыв коммуникационных барьеров, 249
программы с открытым исходным кодом, 30, 346
лицензии GPL и BSD, 30
продюсирование музыки, 561
проекты
GNU Emacs, 357
Linux, 167
skunkworks, 343
повышения продуктивности разработки ПО (ППР), 278, 292
выбор технологий, 284
команда, 283
обучение руководства, 287
оценка и отбор вариантов применения спиральной модели, 281
поддержка ПО, 289
сбалансированная команда, 282

стандартизация, 286
окупаемость, 186
по результатам тестирования (в XP), 243
с открытым исходным кодом, 94
коммит-рассылка, 355
определение кандидатов на роль основных специалистов, 347
прозрачность в Agile-разработке, 335
промежуточные версии ПО, 154
профессиональные навыки членов команды, 295
процесс разработки ПО, 441
Candygram, проект, 484, 489
Kung Fu Rag Dolls, игра, 60
Лаборатория реактивного движения (ЛРД), 297
многократное использование кода, 437
препятствия для изменения, 253
работа дома после терактов 11 сентября, 436
резервное копирование исходного кода, 435
требования, 428
унифицированный процесс Rational (RUP), 438
процессоры Intel I960 и AMD 29050, 383
проектирование повторного использования ПО, 155

P

развитие навыков, 107
размер команд, 165
разногласия, 51
разработчики, 83, 94
конфликты, 89
критика и реакция на нее, 87
найм, 91, 93
управление, 91
свободного ПО, 94, 147
социальное взаимодействие, 84, 87
уважение к членам команды, 89, 91
этикет гиков, 88
расписание
контроль в Agile-методологии, 339
распределенные команды, 156, 162
доверие, 163
культурные различия, 163
«проблема курилки», 163
расстановка приоритетов задач проекта, 198
реакция на критику, 87
ревизионный номер, 351
резервное копирование сервера, комплексные проверки, 435
ресурсы и люди, 251
рефакторинг кода, 171
в экстремальном программировании (XP), 243
риск
незаменимый программист, 271
связь с успехом, 110
родовая память, 166
руководство
воспитание в проекте ППР в TRW, 283
команды разработчиков, 91
обучение на проекте, 287
проекты с открытым кодом, 94
руководство командами
забота о каждом участнике команды, 445
плохой босс, 403, 423
Рэймонд, Эрик, 30

C

самоорганизующиеся команды, 447
самоотбор, 215
сбор информации, 167
система учета, 312
системная метафора в XP, 243
системы оперативного управления (военные), 219
согласование сроков, 174
соглашение о вещании, 204
Соглашение о доступе к информации, 210
солидарность, 77
сообщения лога (Contribulyzer), 351
сопротивление переменам, 99
сотрудничество
и открытая среда, 252
команд разработчиков, 150
программирование в парах в XP, 245, 249
улучшение, 254
социальная адаптированность, 87
социальное взаимодействие, 84
в крупных командах, 165
разработчики, которых нужно оставить в покое, 164
социальные потребности, 84
спецификация требований по программному обеспечению, 321, 484

спиральная модель параллельной разработки, 279, 281
сплоченность, 185
стандартизация, 286
стили работы, 165
Стори, Дэвид, 117
стресс, 301

Т

твердой руки, принцип, 34
творческий импульс, 103
  подавление, 241
текстовые редакторы, влияние на производительность, 288
тестирование, 62, 175
  автоматическое, 193
  и защита от ошибок, 298, 299
  методы, 304, 308
  приемо-сдаточные тесты в XP, 243
техническая экспертиза, 325, 326
технические проверки, 241
технический бум 1990-х, 68
технология разработки ПО, 425
  быстрая смена, 290
Тиманн, Майкл, 31
тимбилдинг, 401
Торвальдс, Линус, 28, 31
требования, 311, 330
  в компании со слабо проработанными процедурами, 428
  для локальных потребителей, 250
  для продуктов, зависимых от классов пользователей, 293
  определение классов пользователей, 314
  для экспертной оценки обнаружения ошибок, 324, 328
  к качеству, 321, 322
  к претендентам, 92
  к применению вариантов использования, 318, 321, 324
    деловые правила, 322
    проверка до программирования, 323
    требования по показателям качества, 317
трудные члены коллектива, 73

У

убеждение, 37
уважение к членам команды, 63, 88
  культура сотрудничества, 251
  разработчики, 89, 91
  честность, 564
ужасные команды, 45, 52
  Internet Explorer 4.0, 52
  ваби-саби, 49
  когда ужасное прекрасно, 48
  таланты, 45, 47
унифицированность, 155
Унифицированный процесс разработки Rational (RUP), 161, 438, 344
Уолл, Ларри, 33
управление проектами, 448
  исключения, 386
  концепция, 449
  практики борьбы с плохим управлением, 452
  предвкушение стиля XXI века, 553
  продюсирование музыки, 561, 577
управление командами
  MP3.com, 129, 133, 135
  воодушевление других, 254
  защита команд, 169
  качества руководителя проекта, 563
  ожидания, 39
  отношения с участниками, 540
  работа со сложными сотрудниками, 271
  самоорганизующиеся команды, 448
  собрания, 546
  характеристики хороших менеджеров, 245
уровни абстракции, 169
успех, 35, 146, 449

Ф

философия
  ориентированная на общество, 545
  ориентированная на рабочий процесс, 545
Фрай, Стивен, 64
фракции в командах, 178
функциональная зависть, 249
функциональность, 155
функциональные требования, 321, 323

Х

характер, выработка, 49

Ц

целевой план, 328
цели, 105

воодушевляющие цели и концепции, 262
диалог с командой проекта, 186, 188, 192
команды и цели руководства, 265
определенность, 184
различие у членов команды, 300
формирование, 195

Ч

честность, 170
честолюбие, 47

Ш

Шлоссберг, Эдвин, 28

Э

эго, подпитка, 79
экспертная оценка, 325, 328
неформальный вариант, 325
проекты с открытым исходным кодом, 359
результаты проверки требований, 328
формальный вариант, 326
экстремальное программирование (XP), 240, 254
культура, 251
методы, 242
сотрудничество, 254
требования для локальных потребителей, 250
электронная почта
внедрение, 283
помощь или преграда производительности, 287
эмоции, 87
энтузиазм, 174
эстетическая концепция, 37, 39
этикет, 88
эффективные методы, 257, 263, 275
высокопродуктивные команды
определение, 258
создание, 261, 275
сохранение, 264
командный дух и мотивация, 266, 269
незаменимые программисты и, 271
открытость и ответственность за действия, 274
помощь плохо работающим командам, 274
проблемные участники, 270
программа обучения работе в команде, 267
условия для приятной работы, 259
устойчивое развитие, 258
цели команды и цели руководства, 265

Ю

юмор, 51

Я

языки
Ada, 380
проверка компилятора, 387
FORTRAN, 542
ассемблера, 386
Янг, Джерри, 35

По договору между издательством «Символ-Плюс» и Интернет-магазином «Books.Ru – Книги России» единственный легальный способ получения данного файла с книгой ISBN 978-5-93286-171-4, название «Идеальные команды. Вдохновляющие и предостерегающие рассказы ветеранов тимлидинга» – покупка в Интернет-магазине «Books.Ru – Книги России». Если Вы получили данный файл каким-либо другим образом, Вы нарушили международное законодательство и законодательство Российской Федерации об охране авторского права. Вам необходимо удалить данный файл, атакже сообщить издательству «Символ-Плюс» (piracy@symbol.ru), где именно Вы получили данный файл.