ГЛАВА 1

Миф о гениальном программисте

Поскольку эта книга посвящена социальным проблемам разработки программного обеспечения, рассмотрим предмет, который вам, безусловно, хорошо знаком — это вы сами.

Люди — создания несовершенные, однако прежде чем искать недостатки в коллегах, поищите их у себя. Подумайте о собственных реакциях, поведении и взглядах, это позволит вам стать более успешным и эффективным разработчиком. Вы станете тратить меньше времени на решение проблем, связанных с людьми, и уделять больше внимания созданию отличного программного кода.

Основная идея этой главы заключается в том, что разработка ПО — это командный вид спорта. Чтобы преуспеть в команде инженеров, сосредоточьтесь на ключевых принципах общения — скромности, уважении и доверии.

Перед тем как углубиться в самосовершенствование, давайте понаблюдаем за обычным поведением программистов.

Помогите мне спрятать свой код!

Мы много раз выступали на конференциях по программированию за последние шесть лет. Будучи участниками первого состава команды, которая запустила службу хостинга проектов с открытым кодом в поисковой системе Google в далеком 2006 году, мы получали

много вопросов и пожеланий, связанных с нашим продуктом. Вот несколько типичных вопросов и пожеланий середины 2008 года:

> *Создайте, пожалуйста, возможность скрытия определенных ветвей кода в Subversion на Google Code.*
>
> *Сделайте так, чтобы было можно «спрятать» проект с открытым кодом в начале работы над ним и опубликовать его в уже готовом виде.*
>
> *Я хочу переписать весь код с нуля, не могли бы вы полностью очистить архив проекта?*

Общая тема в этих запросах очевидна, не правда ли?

Ключевая проблема здесь — *незащищенность*. Люди боятся, что другие увидят и подвергнут оценке их незаконченную работу. С одной стороны, это особенность человеческой натуры: никто не любит подвергаться критике, особенно за незавершенный результат. Эта человеческая черта проявилась в виде описанной тенденции при разработке ПО. На самом деле незащищенность является симптомом более крупной проблемы.

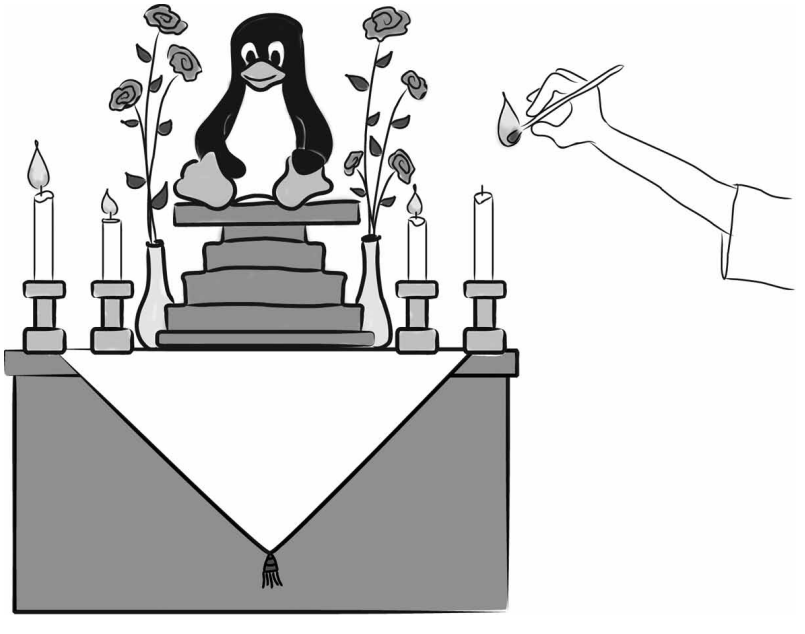
Миф о гении

Для начала скажем прямо: мы не поклонники спорта. Когда наши жены ликуют во время просмотра бейсбольных или футбольных матчей по телевизору, мы чешем затылки, недоумевая по поводу такого накала страстей. Тем не менее в 1990-е годы мы были свидетелями чемпионатов с участием «Chicago Bulls» (это баскетбольная команда, к слову). В то время мы оба жили в Чикаго, и национальные СМИ на протяжении многих лет были наполнены статьями об этой удивительной команде.

О чем же больше всего рассказывало телевидение и писали газеты? Не о команде, а о суперзвезде — Майкле Джордане. Каждый баскетболист в мире мечтал *быть* им. Мы наблюдали, как он «тан-

цует» вокруг других игроков и участвует в рекламных роликах. Мы ходили на дурацкие фильмы, где он играл в баскетбол с картонными персонажами. Он был звездой, и каждый ребенок, кидающий мяч в кольцо, втайне мечтал вырасти и пойти по его стопам.

Программисты подчиняются тому же инстинкту — искать идолов и поклоняться им. Линус Торвальдс, Ричард Столлмэн, Билл Гейтс — все это герои, изменившие мир своими подвигами. Линус ведь создал Linux своими руками, не так ли?



Избегайте естественного инстинкта идолизации вещей

На самом же деле Линус написал лишь прототип Unix-подобного ядра и опубликовал его в списке рассылки. Это была непростая задача, и ее решение, безусловно, впечатляет, однако это всего лишь верхушка айсберга. Операционная система Linux разрабатывалась сотнями квалифицированных людей. Настоя-

щее достижение Линуса в том, что он руководил этими людьми и координировал их работу; Linux же — блестящий результат их коллективного труда (сам Unix был создан небольшой группой специалистов Bell Labs, а не исключительно усилиями Кена Томпсона и Дэнниса Ричи).

Продолжим рассуждение: являются ли все программы Free Software Foundation результатом творчества исключительно Столлмэна? Он написал первое поколение Emacs, однако сотни других людей отвечали за создание *bash*, цепочки инструментов GCC и других программ, работающих на Linux. Стив Джобс руководил целой командой, создавшей Macintosh; главное достижение Билла Гейтса, известного автора интерпретатора BASIC для первых домашних компьютеров, заключается в создании успешной компании вокруг MS-DOS. Тем не менее все эти люди стали лидерами и символами своих коллективных достижений.

А как на счет Майкла Джордана?

С ним — та же история. Мы делаем из него идола, но он не выигрывал все баскетбольные матчи сам. Его гениальность заключается в его работе *со* своей командой. Тренер команды, Фил Джексон, будучи очень умным человеком и талантливым наставником, понимал, что один игрок никогда не выигрывает чемпионат, и поэтому создал целую «команду мечты» во главе с Джорданом. Эта команда была подобна хорошо смазанному механизму и производила не меньшее впечатление, чем сам Майкл.

Так почему же во всех этих историях мы постоянно идолизируем одного человека? Почему люди покупают продукты, рекламируемые знаменитостями? Почему мы хотим купить платье как у Мишель Обама или кроссовки как у Майкла Джордана? Знаменитость всегда на первом месте. Естественный инстинкт людей — искать лидеров и ролевые модели, идолизировать их и пытаться подражать им. Для вдохновения всем нам нужны герои, и такие герои есть и в мире программирования. Феномен «знаменитого технаря» уже почти стал частью мифологии. Мы все хотим создать нечто способное

изменить мир, как Linux, или разработать очередной выдающийся язык программирования.

В глубине души все мы тайно мечтаем быть гениями. Самая большая мечта компьютерного энтузиаста — быть озаренным потрясающей новой идеей. Вы проводите в «Пещере Бэтмена» недели и месяцы, тратя все силы на совершенное воплощение своего замысла. Наконец, вы демонстрируете свои программы миру, шокируя гениальностью всех и каждого. Коллеги в восторге от вашего мастерства. Люди выстраиваются в очередь за вашими творениями. Слава и успех ждут вас за порогом.

Давайте сравним желаемое с возможным. Скорее всего, вы не гений. Мы ни в коем случае не пытаемся обидеть вас и уверены в том, что вы — очень умный человек. Однако понимаете ли вы, как редко встречаются *настоящие* гении? Да, вы пишете код, и, вероятно, это умение ставит вас выше значительной части человечества. Однако, даже если вы гений, *этого недостаточно*. Гении тоже совершают ошибки, и наличие великолепных идей и выдающихся навыков программирования не гарантирует, что ваши программы ждет успех. Условием побед или поражений в карьере является то, насколько хорошо вы сотрудничаете с другими людьми. Оказывается, что «миф о гении» — лишь еще один аспект нашей незащищенности. Большинство программистов боится делиться незавершенной работой, поскольку коллеги, увидев их ошибки, узнают о том, что автор кода — *не гений*. Вот цитата программиста, написавшего в блог Бена:

> *Я знаю, что сильно комплексую по поводу людей, которые видят то, что еще не закончено. Мне кажется, что они строго осуждают меня и считают идиотом.*

Этот комплекс очень распространен среди программистов, и естественная реакция на него — спрятаться в пещеру и работать, работать, работать. Пусть никто не видит ваши промахи: есть шанс исправить их, прежде чем вы представите свой шедевр миру. Спрячьтесь подальше до тех пор, пока все не станет идеальным.

Еще один фактор, побуждающий прижимать карты к груди, — это страх, что другой программист воплотит вашу идею раньше, чем вы сумеете ее реализовать. Держа замысел в секрете, вы контролируете его.

Мы знаем, о чем вы сейчас думаете: «Ну и что? Разве нельзя дать людям возможность работать так, как они хотят?» На самом деле нет. Мы утверждаем, что в этом случае вы поступаете неправильно, и это *важно*. Попробуем объяснить, почему.

Скрываться вредно

Работая в одиночку, вы *увеличиваете* риск неудачи и уменьшаете потенциал роста.

Откуда вы знаете, правильным ли путем идете?

Представьте, что вы увлекаетесь проектированием велосипедов и однажды вам в голову приходит великолепная идея о совершенно новом устройстве переключателя передач. Вы заказываете детали и неделями не вылезаете из гаража в попытке создать прототип. Когда сосед, который тоже увлекается велосипедами, интересуется, чем вы заняты, вы решаете не говорить ему об этом. Вы не хотите, чтобы кто-либо знал о проекте до тех пор, пока он не будет доведен до полного совершенства. Через несколько месяцев выясняется, что вы не можете заставить прототип работать как надо, но обстановка строгой секретности лишает вас возможности получить совет от друзей, разбирающихся в механике.

В один прекрасный день ваш сосед выезжает из своего гаража на велосипеде с совершенно новым механизмом переключения передач. Выясняется, что он занимался тем же, чем и вы, но с помощью друзей из велосипедного магазина. Вы упали духом и показываете ему свою работу. Он говорит, что в вашей модели было несколько простых ошибок, которые можно было устранить в первую неделю, если бы вы показали ему свою модель.

Работа в изоляции приводит к разочарованию

Из этой истории следует извлечь несколько уроков. Скрывая великолепную идею от внешнего мира и отказываясь демонстрировать кому-либо результаты до тех пор, пока они не будут «блестеть», вы очень рискуете. В начале работы легко допустить фундаментальные ошибки в проектировании. Вы пытаетесь заново изобрести колесо, а также лишаетесь преимуществ совместной работы: видите, насколько быстрее двигался вперед ваш сосед, сотрудничая с другими людьми? Именно поэтому люди трогают воду, перед тем как нырнуть в глубину: вам нужно удостовериться, что вы работаете над подходящей идеей, делаете это правильно и никто не сделал этого раньше вас. Вероятность неверного шага в начале пути высока. Чем больше отзывов вы получите в начале работы, тем больше снизите этот риск. Запомните проверенный практикой принцип: «ошибайтесь рано, ошибайтесь быстро, ошибайтесь часто». Мы подробно обсудим важность ошибок позднее в этой книге.

Демонстрация результатов работы на ее ранних стадиях не только проверит идеи и предотвратит ваши ошибки. Но и увеличит то, что мы называем *автобус-фактором* проекта.

Автобус-фактор (сущ.): число участников проекта, которые должны быть сбиты автобусом, чтобы проект потерпел окончательный крах.

Каков автобус-фактор вашей команды?

До какой степени распределены знания и ноу-хау в вашем проекте? Если вы — единственный человек, который понимает, как работает код прототипа, это может быть хорошей подстраховкой от увольнения, но с другой стороны, если проект потерпит крах, то вас «собьет автобус». Работая с другом, вы удваиваете автобус-фактор. При наличии команды, которая проектирует и строит прототип общими усилиями, ситуация еще лучше: проект не закроется с исчезновением одного участника команды. Запомните: если участники команды не попадают под автобус в буквальном смысле, то это не исключает других непредсказуемых событий, происходящих в жизни. Кто-то может жениться или выйти замуж, переехать, уволиться или оказаться вынужденным ухаживать за заболевшим

родственником. Необходимо обеспечивать успешный ход проекта в будущем, управляя его автобус-фактором.

Помимо автобус-фактора, существует также проблема общей скорости продвижения проекта. Легко забыть о том, что работа в одиночку часто бывает трудной и идет значительно медленнее, чем требуется. Сколько нового вы узнаете, работая изолированно? Как быстро вы продвигаетесь? Всемирная паутина наполнена информацией, однако она не заменяет реального человеческого опыта. Работа с другими людьми напрямую увеличивает коллективный опыт, стоящий за усилиями. Когда вы «зависаете» над бессмысленной задачей, сколько времени вы тратите на то, чтобы выбраться на правильный путь? Подумайте, как изменился бы процесс работы, если бы рядом находилась пара коллег, которые могли заглянуть через ваше плечо и без промедления указать на ошибки и способы их исправления. Именно поэтому в компаниях, разрабатывающих ПО, участники команд сидят вместе или занимаются парным программированием: вторая «пара глаз» часто оказывается необходимой.

Вот еще одна аналогия: задумайтесь о том, как вы работаете с компилятором. Создавая большую программу, нажимаете ли вы кнопку «компилировать» в первый раз через несколько дней или недель написания кода, когда уверены в том, что все готово и идеально? Разумеется, нет. Представьте, какая неудача ожидает вас при попытке скомпилировать 50 тысяч строк кода с нуля! Как программисты, мы лучше всего работаем в *коротких* циклах обратной связи: пишем новую функцию и компилируем, добавляем тест и компилируем, реорганизуем код и компилируем. Сгенерировав код, мы исправляем ошибки и опечатки как можно раньше. Компилятор подстраховывает нас на каждом маленьком шаге; некоторые среды разработки способны выполнять компиляцию в то время, *когда мы набираем код*. Таким способом мы поддерживаем высокое качество кода и гарантируем корректную и поступательную разработку нашего ПО.

Такая же обратная связь требуется не только на уровне кода, но и на уровне проекта в целом. Амбициозные проекты развиваются быстро и должны «на марше» адаптироваться к изменениям внешней среды. По ходу развития проекта могут возникать непредсказуемые препятствия при моделировании, политические проблемы и просто ситуации, когда все работает не так, как планировалось. Иногда неожиданно изменяются требования. Как обеспечить обратную связь, которая в нужный момент подскажет, что планы или модели нуждаются в корректировке? Ответ: работая в команде. В связи с этим часто цитируют высказывание Эрика Рэймонда «много глаз видят все ошибки», однако, возможно, лучшая интерпретация звучала бы так: «много глаз обеспечивают проекту адекватность и правильное направление». Люди, работающие в «пещерах», внезапно обнаруживают, что их исходное миропонимание, возможно, и является полным, но сам мир изменился и их продукт перестал быть актуальным.

ИНЖЕНЕР И ОФИС

Двадцать лет назад бытовало мнение, что продуктивный инженер обязательно должен иметь отдельный кабинет с закрытой дверью. Считалось, что это единственный способ обеспечить инженеру спокойный творческий процесс, при котором он мог сконцентрироваться на написании большого количества программного кода.

Мы полагаем, что большинству инженеров не только необязательно, а опасно работать в отдельной комнате[1]. Сегодня ПО разрабатывается коллективами, а не отдельными людьми, и постоянная оперативная связь разработ-

[1] Тем не менее мы согласны с тем, что ярко выраженным интровертам, как правило, требуется проводить больше времени в тихой, спокойной и уединенной обстановке, чем большинству людей, и они чувствуют себя лучше если не в отдельной комнате, то хотя бы в более спокойном месте.

чика с коллегами важнее, чем непрерывное подключение к Интернету. Можно работать хоть 24 часа в сутки, но двигаясь в неверном направлении, вы просто теряете время. Зайдите в офис любой быстроразвивающейся высокотехнологичной компании, созданной в XXI веке, и вы увидите, что инженеры сидят группами в кабинках или за общими столами; случаи, когда инженеры изолированы друг от друга в отдельных кабинетах, встречаются редко.

Разумеется, при таком подходе необходимо бороться с шумом, отвлекающим людей от работы. В большинстве известных нам команд были придуманы способы, демонстрирующие окружающим, что люди заняты и не хотят, чтобы их отвлекали по пустякам. Мы работали с командой, использовавшей голосовой протокол обращений: если вы хотели поговорить с кем-либо, вы произносили фразу «запрашиваю Мэри», где Мэри — имя того, кого вы хотели отвлечь. Если Мэри могла прервать работу, то она поворачивала кресло и выслушивала вас. Если она была слишком занята, то отвечала «принято», и вы могли заниматься другими делами до тех пор, пока она не закончит текущую работу.

В других командах участникам раздавались шумоизолирующие наушники. Во многих компаниях надетые наушники являются сигналом, означающим «не беспокоить, если вопрос не безотлагателен». В некоторых командах есть символы или плюшевые игрушки, которые участники вешают на мониторы, чтобы обозначить, что их следует отвлекать только при крайней необходимости.

Не поймите нас превратно: мы убеждены, что инженерам нужно время для сосредоточенной работы, когда они могут сконцентрироваться на написании кода, однако еще больше им нужна постоянная и плодотворная связь с командой.