Содержание

Глава 1

Как привыкнуть к роли руководителя

Приступая к новым для нас обязанностям, мы всегда, с одной стороны, питаем большие надежды, а с другой — испытываем определенный страх неудачи. Как сформировавшийся программист вы, конечно, успели приобрести опыт участия в проектах и работы в компаниях. Теперь, получив должность руководителя группы программистов, вы столкнулись с новой и, наверное, слегка обескураживающей задачей. Чтобы преуспеть в неизвестной доселе роли в процессе разработки программного обеспечения, вы должны как можно быстрее пройти путь от программиста до руководителя. При этом вам придется приспособиться к новым общественным условиям и новым механизмам взаимодействия — как с людьми, так и с процессами.

Как известно, совершить рывок чистой физиологии к одухотворенному состоянию человеку помогла адаптация — ведущая сила биологической эволюции. Этот процесс продолжался многие миллионы лет, но результат налицо — люди говорят на разных языках и оперируют абстрактными понятиями вроде компьютерных программ. Так как же мы до этого докатились? Вообще-то этот вопрос лучше задать биологу, но из содержания этой книги вы убедитесь, что адаптация значительно упрощает задачи, стоящие перед руководителями программистов.

Речь в этой главе пойдет о людях. Мы поговорим о самом что ни на есть человеческом занятии — написании кода. В частности, мы разберемся в психологии людей, посвящающих себя этому замечательному занятию. Чем лучше вы знаете людей, которыми вам предстоит руководить, тем выше шансы на успех. Принципов и методик руководства программистами сегодня развелось великое множество. Каждое поколение руководителей, естественно, отталкивается от собственного опыта и устраивает свою деятельность исходя из известных стилей руководства и менеджмента. Лично я принадлежу к тому поколению, которое привыкло орудовать логарифмическими линейками и перфокартами, и, конечно, это обстоя-

тельство накладывает некоторый отпечаток на мои рассуждения. Тем не менее, проработав много лет с программистами значительно моложе себя, я понял, что подходы, свойственные моему поколению, отнюдь не уникальны. Не раз сталкиваясь с трудными задачами руководителя, я старался привыкать к новым направлениям бизнеса, к технологическим изменениям, да и просто боролся с собственным упрямством. Этим своим опытом я намерен поделиться с вами, и очень хочется надеяться, что вместе мы проведем замечательное исследование этого вопроса.

Правда ли, что настоящие руководители ходят в черном?

Некоторые — ходят. Иные даже носят на голове хвосты (хотя это, конечно, зависит от того, сколько у них осталось волос). Так они пытаются повысить свой авторитет в глазах подчиненных им придурков или «ботаников» — выберите тот эпитет, который вам больше нравится. Вполне возможно, вам не нравятся оба варианта, а себя вы ощущаете руководителем современного типа, организующим подобных вам специалистов, которые искренне считают программирование пищей для ума. Что я хочу сказать? Стереотипы (в том числе упомянутые в заголовке раздела) не следует воспринимать слишком серьезно. О чем действительно стоит задуматься, так это о личных взаимоотношениях с программистами и своем месте среди них. Став руководителем, вы уже не имеете права вести себя так, как вели, будучи одним из них. Кроме того, ваше положение шефа в процессе разработки программных продуктов иногда оказывается очень кстати. Как кто-то когда-то сказал: «Дайте мне длинный рычаг, и я поверну Землю». Руководство — это как раз такой рычаг.

Вполне допускаю, что мои пассажи насчет несущественности имиджа вас не убедили. Знаете, я сам довольно долго подходил к мысли о том, что мои внешние атрибуты не обязательно отражают мою внутреннюю сущность. Мне до сих пор нравится стиль «ботаника», но теперь я знаю, что для руководства моей группой этого совершенно недостаточно. Иной руководитель чуть ли не полностью посвящает себя убеждению своих подчиненных в том, что он до сих пор один из них. Но так ли это важно? Вспомните фильм «Сеть» (The Net). Сетевые приятели его главной героини Анжелы (в исполнении Сандры Баллок) признали ее единомышленницей и с готовностью приняли в свой странноватый круг общения. Только вот в конце концов выяснилось, что ничего особо замечательного в этом кругу нет. Отсюда урок: образ человека не может быть поверхностным. Что действительно имеет значение, так это характер. Почему стандартные методики менеджмента так часто на практике оказываются несостоятельными? Да просто потому, что их последователи, вместо того чтобы выработать собственный индивидуальный стиль, забивают методиками свои головы и действуют по ним как по шаблону.

Насколько важно быть крутым?

Итак, если мы заговорили в таком серьезном тоне, стоит ли все-таки постоянно ходить в черном и эксплуатировать стереотипы, которые, по вашему мнению, определяют уровень руководителя программистов? Чтобы оценить, насколько вы крутой, рекомендую пройти тест (см. ниже). Кстати, имейте в виду,

что некоторые предпочитают термину «крутой» слово «нинджитсу»[1], но я придерживаюсь традиционных понятий.

Не забывайте, что моя книга предполагает некоторую работу над собой, — так что не слишком расслабляйтесь. Неожиданные проверки не являются прерогативой старых и противных университетских профессоров — в реальной жизни они подстерегают нас постоянно.

НАСКОЛЬКО ВЫ КРУТОЙ?

Выберите один или несколько вариантов ответов на следующие вопросы.

1. «Хакер» — это тот, кто:
   а) вырубает мебель топором;
   б) не теоретизирует, а увлеченно программирует;
   в) удовлетворяет свои интеллектуальные амбиции, творчески преодолевая или обходя препятствия;
   г) руководствуясь злым умыслом, пытается похитить секретную информацию;
   д) персонаж, сыгранный Анжелиной Джоли.
2. «Взломщик» — это:
   а) человек, который взламывает системы безопасности компьютеров;
   б) белый парень с Юга, вроде вашего автора;
   в) нечто, еще более неуловимое, чем cookie (см. вопрос 6);
   г) латентный хакер.
3. «Фрикинг» — это:
   а) искусство и техника взлома телефонных сетей;
   б) старый «ботаник», строящий из себя крутого.
4. «Пинг» — это:
   а) отправка интернет-пакетов;
   б) писк ультразвукового прибора;
   в) начальная часть названия игры «пинг-понг»;
   г) много счастья, и все сразу.
5. «Червь» — это:
   а) оптический диск с однократной записью и многократным считыванием;
   б) программа-вирус, разрушающая данные в памяти или на диске;
   в) двусторонне симметричное беспозвоночное.
6. «Cookie» — это:
   а) маркер доступа, который передают друг другу взаимодействующие программы;
   б) нечто, ставшее известным благодаря Амосу;
   в) нечто, в чем хранятся (а иногда — анализируются) данные о поведении пользователей при посещении ими веб-страниц.

[1] Вы ведь знаете, что такое «ниндзя», правда? Это слово обозначает исключительность. Получается своего рода «черный пояс в программировании».

Ну как, справились? Однажды мне пришлось читать лекцию по компьютерной безопасности студентам, имевшим к программированию весьма отдаленное отношение. В тот момент я преследовал единственную цель — составить характеристику людей, которые, во-первых, занимаются хакерством, во-вторых, защищают компьютерные системы от потенциальных угроз. Они не слишком хорошо справились с заданием — держу пари, у вас получилось значительно лучше. Дело в том, что все варианты ответов на все вопросы правильны[1] — ну разве что вариант б ответа на вопрос 3 я выдумал. Но на самом деле этот тест успешно подтверждает то обстоятельство, что традиционно программистов приписывают к определенной субкультуре. Иногда ее называют (да не обидятся на меня специалисты) «хакерской» культурой (хотите убедиться? взгляните на варианты ответов на вопрос 1 еще раз — они довольно забавны, не правда ли?). Сегодня стереотипы, связанные с хакерством, понемногу уходят. Даже начинающий программист в сегодняшних условиях — это, как правило, выпускник колледжа или университета, да еще к тому же обладатель магистерского диплома по экономике и управлению. В то же время в каждой компании своя культура, и группа, которой вы руководите, — не исключение. Она в целом не менее уникальна, чем каждый из работающих в ней специалистов. Вне зависимости от определения культуры, она в любом случае совпадает с контекстом ваших обязанностей как руководителя программистов. Стоит хотя бы немного разобраться в культуре ваших подчиненных (в частности, в том, как они мыслят и выстраивают свое взаимодействие), и качество ваших отношений, равно как и эффективность исполняемых вами руководящих функций, сразу возрастет. Так что, если очень хочется, вы, конечно, можете носить черную футболку с таинственным посланием на груди, но имейте в виду, что, подстраиваясь под стереотип руководителя и всемерно подкрепляя его собственным примером, вы сознательно выбираете далеко не самый эффективный стиль управления. Значительно более эффективным механизмам руководства как раз и посвящена эта глава.

Стереотипы, связанные с хакерством, понемногу уходят. Даже начинающий программист в сегодняшних условиях — это, как правило, выпускник колледжа или университета, да еще к тому же обладатель магистерского диплома по экономике и управлению.

Мало быть крутым — смотри в оба!

Конечно, если вы сами отъявленный хакер, проблем по части общения с программистами у вас не возникнет. Тем не менее возьму на себя смелость вас предостеречь: становясь руководителями, даже самые крутые программисты не всегда идеально справляются со своими новыми обязанностями. У них возникает непреодолимое желание самим работать над проектами, которые, по идее, нужно делегировать подчиненным. Они тратят уйму времени на обзоры кода, хотя на это хватит и часа. Они пытаются вылизать все комментарии и отступы. Иногда они бросают попытки объяснить окружающим, что они от этих окружающих хотят, и пытаются все

[1] Большинство этих терминов разъясняются в издании The New Hacker's Dictionary, Third Edition, by Eric S. Raymond (The MIT Press, 1998).

делать сами. Я хочу, чтобы вы меня правильно поняли: вы действительно должны держать в голове как можно больше подробностей, касающихся разработки кода, за который несете ответственность, но также вы должны уметь мыслить глобально, чего, к сожалению, программисты-менеджеры зачастую делать не умеют.

Вы должны держать в голове как можно больше подробностей, касающихся разработки кода, за который несете ответственность, но также вы должны уметь мыслить глобально, чего, к сожалению, программисты-менеджеры зачастую делать не умеют.

Бывает и другая крайность. Некоторые менеджеры днем руководят, а ночью пишут код, — как вы понимаете, ни на что другое у них не остается времени. В таком случае кодирование воспринимается как главная ценность в жизни, а руководство — как неприятная обязанность. Такая схема, в принципе, срабатывает, но лишь до тех пор, пока не пропадает всякое желание работать. С моей точки зрения, любой профессиональный руководитель программистов обязан лелеять в себе страсть к работе, не давать этой страсти исчезнуть. В этой и нескольких следующих главах мы рассмотрим ряд приемов, к которым менеджерам рекомендуется обращаться, чтобы гармонизировать свою работу и в то же время не потерять желания трудиться. Один из таких приемов, позволяющий выделять время на выполнение руководящих функций, — делегирование. Поскольку делегирование есть краеугольный камень всякой руководящей деятельности, ему полностью посвящена глава 8. Пока что вы должны четко уяснить себе, что делегирование невозможно без доверия к своим подчиненным. Для того чтобы построить доверительные отношения, требуется некоторое время, но без них деятельность руководителя обречена на провал. Не следует также забывать, что доверие предполагает взаимность. Мне очень хочется, чтобы, прочитав эту главу, вы научились доверять своему интуитивному представлению о людях и путем некоторого анализа интеллектуальных способностей и личностных характеристик своих программистов смогли лучше в них разбираться.

Как руководить чокнутыми, чудаковатыми, странными и обычными программистами

Не хотелось бы говорить об управлении программистами исключительно в ироническом тоне — хотя надо заметить, что этот вид деятельности кто-то очень удачно сравнил с выпасом котов, имея в виду, конечно, творческий характер людей, занятых написанием кода. Проблема в том, что управлять этими иногда беспокойными, неизменно нужными и, как правило, очень интересными сотрудниками крайне трудно. Чем лучше вы будете их знать, тем совершеннее станет ваш стиль руководства.

Если вы программист-эстет, то выражение «быть на короткой ноге с кодом» вам, конечно, известно, — код в таком случае оказывается чуть ли не второй натурой программиста. Элен Алман (Ellen Ullman) в своей книге «Close to the Machine» выражается по этому поводу следующим образом.

«Один из моих знакомых руководителей проектами однажды сравнил процесс управления программистами с выпасом котов. Он хотел сказать, что песики,

преданно заглядывающие в глаза, нам совершенно не нужны. Хорошего программиста нужно ценить вместе со всеми его странностями. С другой стороны, всех этих хороших программистов нужно каким-то образом заставлять двигаться в одном направлении»[1].

Вот это «одно направление» отражает задачу, которая стоит перед каждым руководителем проекта. Но ведь двух одинаковых программистов не существует, и поэтому для каждой группы специалистов нужно выбрать индивидуальный стиль руководства. Управлять программистами невозможно, если в них не разбираться. Поэтому в следующем разделе я привожу перечень характерных «типов» программистов и для каждого из них обозначаю отличительные характеристики. Скорее всего, в них вы узнаете кого-то из своих подчиненных. Поскольку наша книга о котах, типы я называю «породами».

Какие бывают породы программистов

На что похож типичный программист? Можно ли построить шаблон программиста, хотя бы для того, чтобы понять мотивы его поведения? Может быть. В психологической науке существуют так называемые тесты оценки личности. Здесь тот же принцип — достаточно разобрать отличительные черты программистов в отдельности, и вы поймете, что многие из этих характеристик (даже если они на первый взгляд кажутся взаимоисключающими) могут существовать в одном лице. Я разделяю породы на три категории: распространенные, редкие и дворовые. *Распространенные* породы встречаются чаще всего. *Редкие* породы встречаются не так часто, но иногда с ними все-таки приходится сталкиваться. *Дворовые* породы, как и следует из их названия, приносят не слишком много пользы, но знать о них нужно, хотя бы в силу того, что они существуют. Если регулярно помогать дворнягам повышать уровень знаний и, соответственно, преодолевать слабости, которые они по определению привносят в процесс кодирования, они могут стать очень достойными исполнителями.

Как я уже говорил, любой отдельно взятый человек может заключать в себе характеристики сразу нескольких пород. Конечно, разобраться в таких людях труднее, но это того стоит. У программистов на редкость богатое «наполнение». Различия между породами и индивидуальные стили, характерные для каждой из них, как мне кажется, нужно смаковать. Вполне возможно, что в следующий раз вы столкнетесь с этими характеристиками, невзначай взглянув в зеркало.

Распространенные породы

Ниже я перечисляю распространенные породы и их характеристики.

Архитектор[2]**.** Большинство руководителей обожают этот тип программистов — и, действительно, любой такой деятель окажется ценным приобретением для вашей команды. В основном архитекторы концентрируются на общей структуре кода. Они мыслят объектами, а их лучший друг — лист белой бумаги. Посвящая себя без остатка решению бизнес-задач, они строят абстракции, проводят

---

[1] Ellen Ullman, Close to the Machine (San Francisco: City Lights Books, 1997), p. 20.

[2] Термином «архитектор» я в данном случае обозначаю один из личностных типов программистов и совершенно не имею в виду полноценного программного архитектора. О том, какую роль играет архитектура в общей схеме разработки, говорится в главе 6.

анализ систем, после чего переходят к кодированию конкретных решений. Слов нет — все это очень важные элементы программирования, но для комплексного выполнения задач их еще не достаточно. Зачастую в высшей степени разумные замыслы архитектора воплощаются в настолько общем и непонятном коде, что людей, могущих разобраться в нем и продолжить начинание, просто не находится. Особи, способные генерировать удачную идею в голове (а лучше в Visio), а затем выполнить ее полноценную конкретизацию в коде, становясь, таким образом, единственными участниками процесса, встречаются очень редко. Недостаток архитекторов в том, что их код часто служит только одному хозяину, а исполнять чужие команды категорически отказывается[1]. Некоторые архитекторы очень любят набросать структуру кода, с тем чтобы впоследствии передать его на растерзание программистам более «низкой» квалификации. Иногда в коде, написанном архитекторами, встречаются весьма странные конструкции — например, окна с сообщениями о системных прерываниях из-за ошибок, появляющиеся по той лишь причине, что код предполагалось исполнять в виде библиотеки DLL на сервере.

Конструктивист. Конструктивисты получают удовольствие от процесса написания кода и его результата. Стратегическим планированием они себя утруждают не всегда, но факт в том, что с написанием кода они справляются быстро, причем в большинстве случаев ошибок в нем не обнаруживается даже на этапе альфа-тестирования. Код конструктивисты пишут по наитию, а потому их логика не всегда понятна. У некоторых конструктивистов все в порядке и с интуицией, и со стратегическим планированием, поэтому код выступает естественным продолжением хода их мыслей. Но стоит попросить конструктивиста составить документацию, он обязательно ответит, что код самодокументируемый. Впрочем, если на него немного надавить и дать понять, что без документации никуда не деться, он, вероятно, согласится ее составить — и сделает это качественно. Кто спорит — код должен быть самодокументируемым, но следует иметь в виду, что основное внимание программисты этого типа уделяют процессу создания кода, поэтому остальное для них не так уж важно. Количеству сборок, которое конструктивист выдает за день, позавидует даже Microsoft. Соответственно, их код обычно отличается надежностью. Однако же по мере разбухания (а этот процесс неизбежен) надежность улетучивается, а конструктивист начинает судорожно искать новые, «заплаточные» решения — ведь для него очень важно видеть результат и пребывать в уверенности в том, что он справился с поставленной задачей. Конструктивист в сочетании с архитектором имеют все шансы стать прекрасной командой. Если же вы умудритесь отыскать конструктивиста и архитектора в одном лице, считайте, что львиная доля кадровых проблем решена.

Художник. На самом деле, искусства в написании кода не меньше, чем науки, — не зря же университеты часто сводят оба направления в одной структуре и называют ее как-нибудь вроде «факультета свободных искусств и наук». Не будь в программировании художественного аспекта, может быть, оно приносило бы

[1] А ведь это очень важно — согласно авторитетным оценкам, по меньшей мере 70 % стоимости программного обеспечения приходится на сопровождение. См. William H. Brown et al, AntiPatterns: Refactotoring Software, Architectures, and Projects in Crisis (New York: John Wiley & Sons, 1998), p. 121.

нам гораздо меньше морального удовлетворения. Художник как тип программиста сконцентрирован на процессе создания кода — переносе коммерческих требований на программные конструкции и искусном сведении объектов пользовательского интерфейса в одну изящную структуру. Работая с компонентами без видимого интерфейса, художники обнаруживают тенденцию к правильной и логичной организации. Недостаток художника в том, что очень часто он затягивает кодирование, пытаясь выяснить, сколько знаков равенства можно установить в одной строке, не нарушив при этом правильность результата булева оператора. С другой стороны, если программист не культивирует в себе художника, результаты его деятельности зачастую отрываются от реальности, теряют «изюминку». Стоит отнять у художника все его отличительные качества, и в результате получится мина замедленного действия, которая обязательно взорвется под пальцами пользователей. Разделяя некоторые характеристики конструктивистов и архитекторов, художники активно претендуют на собственный стиль.

Инженер. Инженеры вам понравятся. Эти ребята имеют обыкновение скупать все возможные средства сторонних производителей, писать десятки COM-объектов и сводить их воедино, так что они прекрасно работают в версии 1. Присущая им тяга к усложнению проявляется лишь тогда, когда речь заходит о версии 1.1. Программирование часто приравнивают к инженерии программных средств — и, действительно, многие стороны нашей профессии подчиняются этой методологии. Но давать инженерам карт-бланш нельзя. В программных продуктах, выстроенных инженерными методами, нет ничего предосудительного — в конце концов, согласно классическому определению, инженерия есть научные принципы, задействованные при решении программных задач. Нам нужны программисты, которые не боятся сложностей, но те из них, которые любят усложнять все и вся, представляют серьезную опасность. Поймите меня правильно: я совершенно не собираюсь бросать камень в огород инженеров. В конце концов, я сам многие годы трудился над аппаратным обеспечением компьютеров. Но аппаратная направленность иногда входит в противоречие с теми аспектами программного обеспечения, благодаря которым оно становится программируемым (то есть гибким и многократно используемым). Любое аппаратное устройство обычно служит одной, четко определенной цели, а для программного обеспечения такой подход неприемлем.

Ученый. Ученые — это мальчики и девочки, которые считают себя последователями Бэббиджа и Тьюринга. Никогда в жизни они не вставят в код инструкцию GoTo. Отодвигая художественную составляющую программирования на второй план, они делают все в соответствии с фундаментальными принципами компьютерных наук. И как раз в этом обычно и заключается проблема. В то время как они одержимы безупречностью своих трудов, ваша главная забота как руководителя состоит в том, чтобы разработать доброкачественный продукт и сдать его к установленному сроку. Программисты такого типа на самом деле очень полезны, и когда речь заходит об особо трудных задачах кодирования, их идеям нет цены. Вы просто должны следить за тем, чтобы их педантичность не перевесила практические соображения. У инженеров и ученых есть одна общая черта — те и другие очень любят все усложнять. Иногда даже кажется, что они все поклоняются богу сложности (и даже приносят ему жертвы!).