БИБЛИОТЕКА ПРОГРАММИСТА

Дж. Ханк Рейнвотер

Как пасти котов

Наставление для программистов, руководящих другими программистами

- Как вести за собой тех, кто привык гулять сам по себе
- Как стать лидером
- Как выявлять и устранять возможные проблемы
- Как общаться с топ-менеджерами

Apress®

ПИТЕР®

Библиотека программиста (Питер)

Дж.Ханк Рейнвотер

Как пасти котов. Наставление для программистов, руководящих другими программистами

«Питер»
2002

УДК 658.014.1
ББК 65.290-23

Рейнвотер Д.

Как пасти котов. Наставление для программистов, руководящих другими программистами / Д. Рейнвотер — «Питер», 2002 — (Библиотека программиста (Питер))

«Как пасти котов» – это книга о лидерстве и руководстве, о том, как первое совмещать со вторым. Это, если хотите, словарь трудных случаев управления IT-проектами. Программист подобен кошке, которая гуляет сама по себе. Так уж исторически сложилось. Именно поэтому так непросто быть руководителем команды программистов. Даже если вы еще месяц назад были блестящим и дисциплинированным программистом и вдруг оказались в роли менеджера, вряд ли вы знаете, с чего надо начать, какой выбрать стиль руководства, как нанимать и увольнять сотрудников, проводить совещания, добиваться своевременного выполнения задач. В таком случае без этой книги вам не обойтись. А может быть, вы – опытный менеджер, желающий пересмотреть свои принципы лидерства? Тогда, опять же, эта книга для вас. Вне зависимости от возраста, пола и социального статуса, она поможет вам укрепить свои позиции в роли лидера программистов. Материал изложен довольно компактно и легко укладывается в голове. Стоя в книжном магазине и раздумывая, что же купить, задайте себе один простой вопрос: «Нужно ли мне совершенствовать свои лидерские навыки?» Полагаю, вы ответите: «Да», – а значит, данная книга окажется для вас небесполезной.

УДК 658.014.1
ББК 65.290-23

Содержание

Дж. Ханк Рейнвотер
Как пасти котов. Наставление для программистов, руководящих другими программистами

J. Hank Rainwater
Herding Cats: A Primer for Programmers Who Lead Programmers

* * *

Посвящается Дэвиду, моему любимому сыну, – память о тебе меня неизменно вдохновляет.

Жаль, что тебя нет, и я никогда больше не увижу, как ты смеешься…

Предисловие

Прочитав замечательную книгу Хэнка, в которой он рассказывает о выпасе котов, я вспомнил то время (а было это… ну очень давно), когда из программиста меня перевели в менеджеры. Подобно вам, читатели, я был в высшей степени самоуверенным программистом-аналитиком. Я специализировался на языке PLI и базах данных IMS DB/DC. Прибавить к этому понемногу Ramis, FOCUS, Easytrieve Plus, Datacom/IDEAL, CICS, VSAM – и получится вполне сформировавшийся программист, пишущий для мэйнфреймов. Сегодняшние кодировщики могут с полным правом относить эти технологии к древней истории, но, смею вас уверить, в те времена по крайней мере некоторые из них были очень даже ничего!

Подобно Хэнку, я сначала взял на себя неофициальные обязанности координатора и наставника своих коллег – в основном, молодых сотрудников. Впоследствии эти полномочия были закреплены за мной уже формально. Таким образом, я начал сочетать полную ставку программиста-аналитика с полноценным руководством. После этого мне открылись как положительные, так и отрицательные стороны менеджмента. Помню, собрались мы – я и еще несколько таких же менеджеров – как-то раз на совещание с начальником. Начальник говорил о том, какие у него на наш счет большие ожидания, о необходимости повышать нашу квалификацию как руководителей. Одна из моих коллег в ответной речи выразила крайне распространенную среди молодых менеджеров позицию. Она заявила, что могла бы стать значительно лучше как руководитель, будь у нее в подчинении более приемлемый персонал.

Не всегда понятно, почему некоторые вещи крепко оседают в памяти на долгие годы, и как раз ее фразу я никак не могу забыть. Полагаю, будь в нашем распоряжении тогда учебник вроде того, что написал Хэнк, переход к менеджерским обязанностям прошел бы значительно менее болезненно. В конце концов, кто мы такие были? Программисты, которым поручили координировать деятельность других программистов. В результате былые приятельские отношения испарились – будто их никогда и не было. Мне совершенно не хотелось менять свое отношение к коллегам, но без этого контролировать их поведение я не мог. Мы остались друзьями, но эти отношения перешли на другой уровень, что ли, и назад пути уже не было.

Сегодня, по прошествии многих лет, я счастлив, что в роли лидера команды разработчиков, руководителя проектов, руководителя группы, руководителя отдела и директора мне приходится координировать действия более чем 200 людей. За счет посещения разного рода курсов и семинаров мне удалось усовершенствовать навыки руководителя. Наконец, слава богу, что пользу от чтения книг по менеджменту я осознал довольно рано. В конце концов, личность в сегодняшних условиях – это опыт плюс прочитанная литература.

Мой опыт перехода из программистов в руководители позволяет в полной мере оценить предложения, высказанные Хэнком в его книге. Его стараниями любой специалист, находящийся в аналогичном положении, может рассчитывать на существенную помощь. Уже в первой главе Хэнк попадает в десятку утверждением: «то, что делаешь ты, не обязательно буду делать я». В самом деле, не с этим ли связаны все те разочарования, которые мы испытываем в период адаптации к роли руководителя? Если вы принимаете эту проблему близко к сердцу, поверьте мне – Хэнк поможет вам преодолеть подобного рода затруднения.

Попробую перефразировать мою бывшую коллегу: заниматься менеджментом было бы значительно проще, если бы все подчиненные были как две капли воды похожи на своего начальника. К счастью, это не так. Люди руководствуются разными мотивами, у них разный уровень знаний, и понять, что движет тем или иным деятелем, не так-то просто. Различия не превозносят одного человека над другим – просто все мы разные. Что делает руководитель? Он координирует и ведет всех этих «котов», которые гуляют сами по себе. Понимать, как коты себя

ведут и как общаются между собой, совершенно необходимо – иначе эффективного лидерства не получится.

Вспоминается мой разговор с начальником о трудностях, причиной которых стал еще один руководитель из числа бывших программистов. Начальник тогда заявил мне буквально следующее: «Том, пока я сижу на этом месте, никто из программистов больше не станет менеджером!» Где-то через год, во время моего отчета перед новым начальником, мы принялись обсуждать одного из технических руководителей, которому удалось добиться поразительных успехов по части организации и мотивирования своих сотрудников. Этот начальник резюмировал свои соображения так: «Том, я думаю, что впредь всех руководителей нам следует набирать из числа технарей».

Эти диаметрально противоположные точки зрения лишний раз доказывают, что нет двух совершенно одинаковых людей. У всех разные таланты, способности, желания и наклонности. Вы, помимо других, должны разобраться в собственных достоинствах и недостатках (см. главу 2) и задействовать свои навыки таким образом, чтобы обеспечить успешную деятельность группы в целом (см. главу 3). Программисты, которым приходится впервые брать на себя обязанности по руководству другими программистами, обнаруживают себя на перекрестье профессий. Некоторые на постоянной основе переходят к менеджерской деятельности. Другие склоняются к программированию, поскольку в этой области от них больше толку. Остальных устраивает промежуточная позиция между руководителем и кодировщиком.

Если оставить в этой книге только три первые главы, а все остальное выкинуть, все равно ее стоило бы прочесть (ее объем, конечно, сильно бы уменьшился). Однако и остальные главы тоже весьма интересны, поскольку Хэнк рассматривает в них чрезвычайно актуальные для молодых руководителей вопросы.

С одной стороны, он говорит о том, что теперь у вас есть формальные административные обязанности. Навыки руководства людьми в контексте успешной деятельности компании очень важны, но, с другой стороны, именно административные вопросы обеспечивают плавное вращение коммерческого маховика. Вам предстоит постоянно заниматься поиском информации и составлять рецензии на выполненные задания. В рамках подведомственной группы вы находитесь на вершине иерархии управления. Лишь за счет дисциплины и самоорганизации люди справляются с административными функциями. Если эти качества в вашем характере отсутствуют, вы станете слабым звеном административного механизма, и ваш собственный начальник будет вынужден постоянно вас подталкивать. Скажите спасибо Хэнку за объяснение стандартной роли администратора – это объяснение помогает понять, что такое же бремя несут многие ваши коллеги.

Глава 5, посвященная проведению совещаний, затрагивает очевидно недооцененный комплекс приемов. Сама постановка вопроса о продуманной организации совещаний заслуживает уважения. Случалось ли вам посещать совещания, не преследующие конкретной цели и никем не управляемые? (Вероятно, этот вопрос лучше переформулировать так: «Каков процент посещенных вами совещаний, которые проводились подобным образом?») Если случалось, теперь вы знаете, почему так происходит: на этих совещаниях никто не проявил активной руководящей позиции. Если вам удастся при проведении совещаний взять на себя лидерство, сделав их тем самым более продуктивными и сориентированными на конкретные задачи, скажите спасибо Хэнку.

Еще один раздел книги, который мне очень понравился, посвящен отношениям с начальством (см. главу 9). Ориентироваться в данной области очень важно, хотя многие слишком поздно осознают это обстоятельство. Да, действительно, ваш начальник должен руководить вами. В то же время активная роль в выстраивании ваших отношений может принадлежать вам. Занятно, но если вы нацелитесь на то, чтобы успех был достигнут вашими подчиненными и начальством, успех обязательно придет к вам самому.

У меня нет возможности углубляться в содержание всех глав и излагать свои соображения по поводу собранного в них материала – в противном случае мое предисловие грозит превысить по объему саму книгу. (Впрочем, позволю себе одно, последнее замечание: обязательно прочтите разделы о многонациональных и распределенных группах. Крайне ценные сведения!) Скажу лишь, что эта книга может существенно облегчить жизнь тех программистов, которые в один прекрасный день обнаруживают себя на руководящих постах. Аналогия с выпасом котов, по-моему, вполне уместна. У многих видов животных развито стадное чувство, и пасти их, соответственно, не так уж трудно. У меня дома два кота, и я на своем опыте знаю, что у них этот инстинкт не просматривается. Вести в заданном направлении даже одного кота (программиста) – нетривиальная задача. А для того чтобы вести за собой четыре-пять (или дюжину) этих упрямых тварей, требуется предельная концентрация и весьма специфический комплекс приемов. На материале этой книги, я надеюсь, вы сможете ознакомиться с требованиями, которые обычно предъявляют к людям, исполняющим вашу роль, и значительно упростить для себя путь к успеху – по крайней мере, опыт Хэнка к этому располагает!

Том Мокел (Tom Mochal), создатель www.TenStep.com, президент TenStep, Inc.

Об авторе

Хэнк Рейнуотер (Hank Rainwater) в настоящее время работает в Risk Sciences Group (Атланта, Джорджия), где руководит группой программистов, разрабатывающих программные продукты для страховых компаний. Его путь в науке и инженерии насчитывает более трех десятилетий. В разные периоды жизни он занимался программированием на языке Фортран с использованием перфокарт; преподаванием математики в колледже; исследованиями в областях радиоастрономии, систем наведения ракет и телеметрических систем; координацией производства встроенных систем цифрового управления. Как специалист в сфере разработки программных продуктов Хэнк успел поработать консультантом, лектором, программистом и руководителем групп разработки программ для самых разных областей человеческой деятельности. Что касается образования, Хэнк окончил колледж с физическим уклоном и получил диплом университета по специальности «математика и физика». Кроме того, Хэнк – магистр теологии. Он несколько лет был пастором в разных приходах (как в США, так и за рубежом), занимаясь попутно преподаванием теологических дисциплин.

Хэнк убежден, что основным условием успешного руководства программистами является наличие лидерских навыков. До прихода в индустрию разработки программных средств он не понимал истинного значения этого утверждения. С опытом он стал более раскрепощенным, отрастил длинные волосы и теперь получает нескрываемое удовлетворение от общения с творческими личностями, старающимися превратить код в продукты с радужными рыночными перспективами.

Без шевелюры его трудно выделить из уличной толпы.

О научном редакторе

Дэйв Кристенсен (Dave Christensen) в настоящее время трудится на посту старшего системно-технического аналитика в целлюлозно-бумажном отделении корпорации Potlatch (штаб-квартира находится в Клокете, штат Миннесота). В его обязанности входит обеспечение компании конкурентных преимуществ за счет разрабатываемых им веб-приложений. Кроме того, он – президент Proxis Productions (http://www.proxis-productions.com) – консалтинговой компании, специализирующейся на проектировании распределенных корпоративных веб-приложений. В 1995 году, когда компания Proxis Productions только создавалась, Дэйв предполагал заняться компьютерной графикой для видеоигр и других коммерческих предприятий, однако с утверждением графики в Интернете он получил возможность свести свои графические и технические интересы воедино. У Дэйва диплом по английской литературе; кроме того, он прослушал несколько медицинских курсов и занимался в театре в колледже St. Scholastica. Дэйв – счастливый человек: свободное от работы время он проводит с прекрасной женой и двумя изумительными детьми, которые не перестают его удивлять. У них много животных: две собаки и выводок кошек, которые, кстати, тоже поучаствовали в работе над этой книгой. Кроме того, Дэйв – коллекционер редких рыбок и увлеченный реставратор. Он обожает раскрывать в окружающих скрытый потенциал. В этом отношении он успел поэкспериментировать на автомобилях, домах и людях.

Об иллюстраторе

Мелани Уэллс (Melanie Wells) – опытный графический дизайнер с десятилетним опытом работы в самых разных областях: иллюстрировании книг, разработке корпоративных логотипов, создании брошюр, проектировании выставочных стендов, оформлении почтовой атрибутики, журнальной рекламы, каталогов, дизайне упаковки, веб-сайтов, спортивной одежды, товарном дизайне и т. д.

Помимо собственно графического дизайна, Мелани увлекается изобразительным искусством. Вне зависимости от текущего занятия – разработки фирменного стиля или, например, рисования маслом на холсте – она в первую очередь художник.

Благодарности

Огромное спасибо Дэну Эплмену (Dan Appleman) из Apress – прочитав черновик первой главы, он по достоинству оценил замысел книги, в которой, в отличие от многих, не рассматривается ни один из современных языков программирования. Ты молодец, Дэн! Уже много лет ты вдохновляешь меня (и не только меня) на новые достижения, и я надеюсь в этом отношении пойти по твоим стопам.

Карен Уоттерсон (Karen Watterson) – человек, чьи рекомендации в период работы над проектом сыграли решающую роль в формировании многих идей, которые ранее находились в зародышевом состоянии. Твои обширнейшие знания, опыт и письма, приходившие в любое время дня и ночи, очень помогли мне. Спасибо, Карен!

Это мой первый опыт на поприще написания книг. В связи с этим не могу не упомянуть Трейси Браун Коллинз (Tracy Brown Collins) – как руководитель проекта она проявила себя с лучшей стороны. Спасибо тебе за проницательные литературные замечания и за то, что благодаря тебе вся наша команда успешно сработалась.

Дэйв Кристенсен – мой главный научный редактор – помог мне разобраться со стилем изложения. Несколько раз я все же оплошал, но с его помощью многие огрехи удалось ликвидировать. Кроме того, я обширно «одалживал» его комментарии. Спасибо тебе, Дэйв, за недюжинные старания.

Я очень признателен Николь Леклерк (Nicole LeClerc) из Apress за грамматическую правку. Мы, программисты, далеко не всегда оказываемся на высоте, сталкиваясь со сложными синтаксическими конструкциями. Куда лучше мы ориентируемся в хитром и загадочном стиле кодирования. Николь восполнила мои школьные пробелы в знаниях касательно орфографии, пунктуации и других мелочей, за счет которых эту книгу стало значительно проще читать.

Джессика Лендисман (Jessica Landisman) провела долгие часы за аннотированием, корректурой, составлением предложений и комментированием первых черновиков. Как опытному высококвалифицированному программисту ей нет равных, и я действительно чрезвычайно благодарен за оказанную мне помощь.

Отдельное «мерси» Кэти Хейнс (Kathy Haynes), корпевшей над рукописью на разных стадиях ее готовности, причем каждый раз после ее участия текст становился неизмеримо лучше. Для меня она, уроженка Восточной Алабамы, – непререкаемый авторитет по диалекту американского Юга.

Все восторги по поводу графического оформления этой книги прошу перенаправлять Мелани Уэллс – замечательной художнице, работающей с самыми разными материалами. У нее настоящий талант на выражение мыслей в визуальных образах. Благодаря тебе, Мелани, коты вышли – круче некуда!

Наконец, я глубоко признателен своему отцу Джулиусу Рейнуотеру, набросавшему кое-какие из иллюстраций, ныне украшающие страницы этой книги. Инженер, предприниматель, замечательный отец, несомненный образец для подражания, обладатель многочисленных талантов – я перед тобой в неоплатном долгу. Ну а без тебя, мама, я вообще ничего не смог бы сделать – даже написать эту книгу!

От издательства

Ваши замечания, предложения и вопросы отправляйте по адресу электронной почты comp@piter.com (издательство «Питер», компьютерная редакция).

Мы будем рады узнать ваше мнение!

Все исходные тексты, приведенные в книге, вы можете найти по адресу http://www.piter.com/download.

Подробную информацию о наших книгах вы найдете на веб – сайте издательства: http://www.piter.com.

Введение

Правды нет – о ней нам только рассказывают.
Анонимный автор с Юга

В этой книге я намерен рассмотреть весь комплекс задач, стоящих перед руководителем. Мужайтесь: решить эти задачи вполне реально, что бы вам ни говорили. От вас требуется выстроить стройную систему мышления, эдакий гештальт. Что такое «гештальт»? Согласно словарю Вебстера, это «структура со свойствами, которые невозможно получить путем простого сложения ее элементов». Если перевести это определение на язык объектно-ориентированного программирования, которым, полагаю, вы владеете лучше, чем стилем изложения Вебстера, получится следующее: вашей мысленной архитектуре предстоит пройти серьезные преобразования. Унаследовав навыки менеджера, вы должны будете перегрузить типичные параметры мышления новыми типами и значениями – то есть, грубо говоря, испытать полиморфизм собственного характера. За счет этого вы инкапсулируете в своем программистском мозгу совершенно новый вид искусства – искусство лидерства и руководства. Между руководством и лидерством прослеживаются существенные различия. Нужно то и другое, но лидерство все же имеет приоритет – хотя крепкие руководящие навыки, естественно, помогают брать новые высоты в роли лидера.

Своевременность производства и качество программных продуктов вашей компании – теперь в ваших руках, и смею надеяться, что моя книга внесет некоторое разнообразие в вашу деятельность. Нельзя же, в конце концов, проводить все рабочие дни совершенно одинаково!

Структура книги

Посмотрим, какие сведения можно почерпнуть из составивших книгу глав.

Глава 1. Как привыкнуть к роли руководителя.

Для развития лидерских качеств нужны новые приемы – навыков, наработанных в бытность программистом, вам не хватит. В этой главе мы поговорим о том, как адаптироваться к новой должности. Для этого я составил перечень наиболее распространенных личностных типов программистов, которые, естественно, оказывают то или иное влияние на способность управлять процессом разработки и направлять его по заданному курсу. Вам надлежит осознать поразительную вариативность характеров подчиненных, попытаться проанализировать их личные качества и найти к ним индивидуальные подходы. В конце концов, вы же главный – что ж тут плохого?

Глава 2. Как руководить собой.

Здесь вам придется добраться до глубин своего сознания (не бойтесь – это не так страшно) и самолично усвоить принципы руководства. Если вы не научитесь управлять собой, занять лидерскую позицию среди коллег не получится. Как говорил Уинстон Черчилль: «Чем пристальнее мы вглядываемся в прошлое, тем проницательнее становимся, предсказывая будущее». Этот афоризм рекомендую применить к вопросам самоанализа.

Глава 3. Как вести стаю за собой.

Лидерская роль предполагает приобретение новых навыков вдобавок к навыкам чисто программистским. В этой главе дается обзор основных областей деятельности лидера, на которые следует обратить особое внимание. В противном случае вы рискуете, поддавшись внешним влияниям, пойти в неверном направлении, а сотрудники группы, подобно испуганным котам, от вас разбегутся. Мне совершенно не хочется, чтобы вы, как говаривал лорд Байрон, оказались среди тех «немногих, чьи души всплывают после крушения надежд».

Глава 4. Как организовать успех.

Здесь мы прервем на некоторое время дискуссию о взаимоотношениях с окружающими. Повысив уровень личностной организации, вы сможете взять новые высоты по административной части. Кроме того, я советую вам изучить организационную структуру своей компании и изыскать способы повышения эффективности работы. Таким способом вы сможете выделить время на развитие лидерских качеств – иначе говоря, на выполнение своих основных обязанностей.

Глава 5. Как вести совещания.

В бытность программистом вы, вероятно, привыкли не советоваться ни с кем, кроме самого себя. Теперь эту ситуацию придется менять. Никаких более совещаний во время утреннего бритья и любования на красавца в зеркале! Вам предстоит обсуждать дальнейшие действия с себе подобными (разве что не такими симпатичными, как вы) и, что гораздо страшнее, с людьми, которые, как это ни странно, не зарабатывают на жизнь кодированием! В роли лидера на совещаниях от вас потребуется терпение. Не отчаивайтесь и не забывайте слова Леонардо да Винчи: «Нетерпеливость – мать глупости».

Глава 6. Философия и методы технического лидера.

В этой главе я рассмотрю некоторые технические принципы и их философские обоснования. Одно дело принимать технические решения применительно к собственному кодовому заданию, и совсем другое – делать это за весь отдел. Вполне возможно, что вы успели взойти на экспертный уровень в области технологии, но это не отменяет необходимости анализа последствий принятия технических решений в корпоративном масштабе. Здесь мы обговорим вопросы архитектуры, проектирования и критических обзоров кода.

Глава 7. Закат лидера.

Все руководители (не только вы, но и ваше начальство) подвержены влиянию упадочных стратегий лидерства, и иногда мы, к сожалению, этому влиянию действительно поддаемся. Некоторые стили руководства не допускают конструктивного лидерства, а значит, их следует избегать. Здесь я опишу возможные варианты деградации лидерских качеств вследствие принятия неверной стратегии и попутно предложу способы выхода из кризиса.

Глава 8. Восход лидера.

Подобно программным продуктам, которые конструируются на основе надежной архитектуры, лидерские качества культивируются на основе присущих лидеру черт характера. В этой главе я попытаюсь свести все аспекты лидерства воедино. Если перефразировать Эмерсона, «многословие – бедствие для авторов, поощряемое издателями, читателями и книготорговцами». Что важнее, здесь я излагаю базовые принципы успешного лидерства и демонстрирую методы их настройки как необходимое условие профессиональности руководства.

Глава 9. Как ужиться с начальством.

Обратите внимание: глава называется «Как ужиться с начальством», а не «Как руководить начальством», – последнее просто не представляется возможным. Тем не менее налаживать отношения с теми сотрудниками, которым вы подотчетны, следует не менее тщательно, чем с собственными подчиненными. Субординация – это совсем не пустое слово. Здесь мы детально обговорим методы формирования слаженной команды из двух человек: вас и вашего босса.

Глава 10. Слова без песни.

В этой главе раскрываются самые разные, порой не связанные друг с другом, темы, которые не всегда касаются ежедневных обязанностей лидера программистов, но, тем не менее, представляют в контексте выпаса котов немалую важность. Руководство распределенной группой разработчиков, оценка тенденций развития методологий разработки программных средств и некоторые другие темы рассмотрены в этой главе. С ее помощью, надеюсь, вам будет проще превратить хаос в порядок и не сойти при этом с ума.

Послесловие. Снова в плавание…

В заключение я извергну из глубин сознания несколько мудрых наставлений – по крайней мере, в нашем сдвинутом, но оттого не менее прекрасном мире разработки программного обеспечения мои слова, полагаю, сойдут за непреходящую мудрость.

Приложение А. Как ухаживать за живностью – электронный администратор.

В этом приложении речь пойдет о программе электронного администратора, которая упоминается в главе 4. Мне совершенно не жалко делиться с вами кодом – надеюсь, что мои идеи не совсем бесполезны, и с их помощью вы сможете доработать решение под себя. Любой обладатель этой книги, написав секретное слово, сможет скопировать код с сайта книги. Впоследствии, по мере материализации моих идей в коде, у вас появится возможность оценить степень их полезности.

Приложение Б. Как дать скотине в глаз – критический обзор кода электронного администратора.

Основной предмет обсуждения здесь – способы практического применения принципов критического обзора кода, рассматриваемых в главе 6, к программе, описанной в главе 4 и приложении А. Хотя мне приходится оценивать собственный код, я стараюсь соблюдать объективность, указывая на его достоинства и недостатки. Тем самым я намерен продемонстрировать вам, как в реальных условиях осуществляются функции кодового полицейского.

Библиография. Ресурсы для специалистов по выпасу котов.

В библиографии фигурируют все работы, на которые по мере изложения материала я счел нужным сослаться, а также несколько других, заслуживающих внимания с вашей стороны. Некоторым участникам нашей лидерской гонки удалось вырваться далеко вперед, и я намерен вас с ними познакомить.

Алфавитный указатель.

Вы знаете, для чего нужна эта штука. Сюда можно заглянуть в поисках отдельных понятий или имен, чтобы не читать книгу целиком. Примерно по тому же принципу обучаются языкам программирования по справочникам. Они дают представление о синтаксисе, но в отсутствие контекста весьма высока вероятность концептуальных ошибок.

По всему тексту разбросаны небольшие врезки под общим заголовком «Кошачьи разборки». В них я рассказываю реальные истории из жизни руководителей программистов, призванные поучать вас и освещать явно неудачную практику. Эти врезки иллюстрируют многие из представленных в книге принципов. Имена, места, языки программирования и типы приложений, конечно, изменены – с тем, чтобы защитить невинных и скрыть виновных. Эти истории легко читаются, а самое главное, наглядно демонстрируют примеры неверных действий. Похоже, мы действительно лучше всего учимся на собственных ошибках. Здесь же, рассказывая об ошибках, совершенных другими людьми, я пытаюсь уберечь вас от повторения их печального опыта. Вспомним Альбера Камю: «Глупы те, кто считает, что пессимистические мысли порождают отчаяние». За счет законов физики мы способны превратить отрицательный опыт в обнадеживающие перспективы – иными словами, опыт окружающих должен подтолкнуть вас к движению в правильном направлении.

Есть и другие врезки, в которых выделены основные положения текущего текста. Такой вариант верстки поможет вам изрядно сэкономить – теперь вам не нужно разоряться на желтый маркер и выделять ключевые моменты изложения. Хотя нет: прибегайте к маркеру ровно столько раз, сколько потребуется. По-моему, самое важное в том, что эта книга открывает простор для осмысления ваших рабочих функций. Чем больше полезного вы в ней найдете, тем лучше – и мне и вам.

Кому и зачем стоит прочесть эту книгу

От чтения этой книги выиграют программисты, превратившиеся в менеджеров, руководителей групп и, если вам по душе более высокие посты, директоров по разработке программного обеспечения. Если вы руководите относительно небольшой группой программистов (состоящей, скажем, из четырех – семи человек), работаете в мелкой или средней компании, заняты разработкой нескольких проектов одновременно – значит, вы выбрали издание правильно. Если же вы, скажем, собираетесь за 12 месяцев построить очередную глобальную систему бронирования авиабилетов, а в вашем подчинении 100 программистов, вероятно, эта книга не будет соответствовать вашему размаху. В таком случае вам лучше защитить две магистерские диссертации: по управлению проектами и по психологии. Удачи.

Если вы руководите программистами – именно руководите – и чувствуете, что лидерство подменяется простым манипулированием проектами и людьми, вам нужна помощь. Моя книга вам поможет.

Быть может, вы – опытный менеджер, желающий пересмотреть свои принципы лидерства. Тогда эта книга опять же для вас.

Не исключаю, что вы наконец-таки получили повышение, на которое давно рассчитывали или которого, наоборот, опасались. Итак, многолетние занятия по написанию интеллектуального кода и конструированию выдающихся программ принесли свои плоды. Начальство посчитало вас подходящим кандидатом на роль руководителя программистов. Вероятно, в вас несколько меньше лунатизма, чем в коллегах. Маловероятно, но все же возможно, что вы привыкли носить сорочки с воротничками, и это обстоятельство сыграло вам на руку. Или кто-то уволился. Как бы там ни было, добро пожаловать в стройные ряды руководителей групп программистов. Моя книга окажется в такой ситуации весьма кстати.

Вне зависимости от возраста, пола и социального статуса эта книга поможет вам укрепить свои позиции в роли лидера программистов. Она довольно компактна, и материал достаточно легко укладывается в голове. Стоя в книжном магазине и раздумывая, что же купить, задайте себе один простой вопрос: «Нужно ли мне совершенствовать свои лидерские навыки?» Полагаю, вы ответите: «Да», – а значит, моя книга окажется для вас небесполезной.

Стиль и позиция

Литература по менеджменту изобилует различного рода предложениями, рекомендациями и научно подтвержденными методиками решения задач при помощи других людей. В конце концов, именно в этом суть менеджмента: делегируя задания подчиненным, вы должны обеспечить мотивацию и проконтролировать (в лучшем смысле слова) выполнение этих заданий ради достижения общей для всей компании цели. Эта книга – отнюдь не очередной фолиант по дисциплине «менеджмент». Скорее, это выраженная в словах концепция, согласно которой для создания первоклассного программного обеспечения требуется мастерство, а оно нарабатывается преимущественно с опытом. Именно из опыта – от этого верного и преданного учителя – я почерпнул те идеи, которые здесь излагаю. Несомненно, что в моей работе фигурируют принципы, сформулированные в свое время в академической среде. Поскольку, скорее всего, чтением вы занимаетесь в перерывах между телефонными звонками и визитами в соседние комнаты к подчиненным программистам, я постарался излагать материал кратко, в позитивном тоне и с юмором – чтобы хоть как-то развлечь вас посреди рабочего дня[1].

В большинстве случаев я употребляю местоимения в мужском роде. Впрочем, примечание в начале главы 9, надеюсь, прояснит для вас мою позицию в этом деликатном вопросе. Никаких предпочтений я здесь не высказываю – мне кажется, лучше вникать в суть, чем заострять внимание на таких вещах.

В этом своем труде я концентрируюсь на двух основных областях руководства программистами: людях и процессах. Из них люди, разумеется, важнее. При наличии хорошей команды разрабатывать прекрасные программные продукты более чем возможно – даже несмотря на недостатки бизнес-требований. А стоит только подогнать требования, и в сотрудничестве с высококлассными специалистами вы сможете докодироваться хоть до Луны!

[1] Мои шутки, по большей части размещаются в сносках, так что они не будут вас отрывать от основной мысли. Ну вот, например: что значит «обратная совместимость»? Это значит, что новая операционная система с тем же успехом, что и старая, может грохнуть все ваши программы.

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 обстоятельство накла-

дывает 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 «ниндзитсу»[2], 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;

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

• 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.

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[3] – 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

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

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 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 пред-

ставлению 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»[4].

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.

---

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

Распространенные породы

Ниже я перечисляю распространенные породы и их характеристики.

Архитектор[5].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 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[6]. 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 нам гораздо меньше морального удовлетворе-

---

[5] 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.

[6] 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.

ния. 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!). Отдавая должную оценку глубочайшим познаниям ученых и по мере необходимости обращаясь к ним, руководитель не должен допускать их полновластия в вопросах написания кода – иначе они сделают его слишком сложным.

Лихач. Лихачи – это те товарищи, которые делают все быстро. Забывая о комментариях, отступах и соглашениях об именовании переменных, они, тем не менее, умудряются достигать результата очень оперативно – и, что самое замечательное, вплоть до первой неперехваченной ошибки их продукты вполне успешно работают. Иногда такое поведение характерно для молодых программистов, горящих желанием впечатлить вас, – они опрометчиво считают, что оперативность в достижении результата в полной мере соответствует вашим ожиданиям. Признайтесь: мы часто сами выстраиваем у них столь ложное представление, а значит, веди

мы себя по-другому, никаких лихачей не было бы. Наши собственные начальники устраивают совещания, на которых устанавливают контрольные сроки, а потом сообщают их нам. Как мы добьемся выполнения поставленных временных задач – это уже наша проблема. Вспомните, как часто идут разговоры о бессмысленности установления крайних сроков кодирования до окончательного выяснения всех требований! Так вот, вам придется к этому привыкнуть. К сожалению, такова реальность – пользователи и рыночные соображения часто принуждают нас сперва давать обещания, а потом уже приступать к планированию. Именно по этой причине вы читаете мою книгу – вам нужны советы по поводу того, как выжить в динамичном, жестоком и суровом мире разработки программных средств.

Редкие породы

Далее я привожу список редких пород и их отличительных черт.

Волшебник[7]. Каким-то загадочным образом те, кого я называю волшебниками, регулярно решают самые трудные задачи программирования, причем идут такими путями, которые раньше никому и в голову не приходили. Более того – волшебники делают все это вовремя, и иногда у них получаются вполне доступные для понимания программы, которые даже можно сопровождать. Немного волшебства в нашем деле не помешает. Но стоит распустить подобным деятелям руки, и вскоре вы превратитесь из здравомыслящего руководителя работоспособной группы программистов в обычного подмастерье. Кроме того, если вы будете слишком полагаться на волшебника, в один прекрасный день он вас разочарует – в конце концов, постоянно творить чудеса никому еще не удавалось.

Минималист. Несмотря на удивительно скромный объем кода, производимого минималистами, код обычно оказывается очень функциональным. Каждая процедура умещается в редакторе кода на одном экране. Объекты стройны, выстроены четко и недвусмысленно сообщают о своем назначении. Звучит неплохо, не правда ли? В общем, да, только стоило бы учитывать мотивы такого поведения. Ведь иногда они кроются в том, что человеку хочется побыстрее разобраться с текущим проектом и перейти к следующему, который его больше захватывает. Иногда (кстати говоря, эта характеристика распространяется и на архитекторов) минималисты, решив поставленную задачу, быстро теряют к ней всякий интерес, и уж, конечно, при обнаружении в ходе альфа-тестирования каких-либо проблем выказывают устойчивое нежелание их исправлять. Иногда минималисты капризны и очень придирчиво выбирают область приложения своих сил. С сопровождением кода дела у них обстоят хуже всех.

Аналогист. Ну ладно, ладно – слово «аналогист» я взял с потолка. Только не подумайте, что это медсестра, которая делает наркоз перед операцией. Нет – это программист, который не слишком силен в абстракциях, но прекрасно справляется с аналогиями. Во время проектных совещаний аналогисты, постоянно выдумывающие все новые и новые аналогии, способны свести с ума любого. Но при этом нельзя не признать, что, как правило, они очень быстро схватывают суть задачи и в результате создают удобный (в том числе и для сопровождения) код. У некоторых аналогистов есть любимые аналогии, которые они норовят применить ко всем без исключения проблемам разработки программных продуктов. Они воображают компоненты маховиками, а успешно справившись со своей задачей, хвалятся тем, что их код «воспламеняется во всех цилиндрах». Их аналогии всегда привязаны к осязаемым объектам, поскольку, как я уже говорил, с абстрагированием дела у них обстоят неважно. Ну, в общем, вы меня поняли. Посадите аналогиста вместе с архитектором, и если они сразу друг друга не прикончат, скорее всего, у них получится превосходный продукт. Правда, поскольку аналогисты не дружат с абстракцией, создавать объекты с четкими межуровневыми интерфейсами у них получается не всегда. Дело в том, что возможность создания в достаточной мере абстрактного интерфейса объекта – это одно из величайших преимуществ объектно-ориентированного программирова-

[7] Некоторые предпочитают называть программиста этого типа «гуру» или «спецом». А мне больше нравится «волшебник».

ния, и поэтому конкретное мышление иногда мешает успешно справляться с поставленными задачами.

Трюкач. Трюкачи слишком увлекаются разными технологическими трюками. Они постоянно осваивают разные новинки, но результат от этого не улучшается. По правде говоря, всех нас в той или иной степени привлекают забавные технологические приемы. Я вот, например, помню мой первый компьютер. Он был аналоговым, и, поворачивая диски, я переключал ветви в предустановленном аппаратном алгоритме. Эта штука была похожа на гипертрофированную логарифмическую линейку. В общем, я до сих пор люблю забавляться со всякими высокотехнологичными штуковинами. Если вам приходится работать с трюкачами, попытайтесь направить их увлечение игрушками на решение их первоочередной задачи, а именно на производство бизнес-решений. Если им удалось втиснуть на экран, который, как предполагается, будет отображаться с разрешением 800 × 600, 30 разных элементов пользовательского интерфейса, это еще совершенно не означает, что они решили свою задачу в соответствии с реальными потребностями пользователей[8]. Трюкачи, при всех их познаниях в технологии, часто не могут усвоить конечное назначение программы. Полагая, что их функции ограничиваются забавами с разными инструментальными средствами, они отказываются учитывать те аспекты программирования, благодаря которым мы не затрачиваем на сопровождение титанических усилий.

Дворняги

Остановимся теперь на дворовых породах и их отличительных чертах.

Разгильдяй. Что сказать о разгильдяях? Некоторые люди небрежны, и это проявляется в коде, который они создают. Они не обращают внимания на такие мелочи, как правильное написание имен переменных и правила венгерской нотации. Зачастую качественно выполнять свои обязанности им мешают проблемы личного плана. Тому, как пишется эффективный код, их нужно учить. Они любят начать с одного стиля, а через процедуру-другую перейти к новому. Читать их код очень утомительно – иногда поздними ночами его приходится переписывать, поскольку иначе есть риск не успеть к сроку сдачи проекта. Если вы не справились с задачей по их вине, прошу вас: отнеситесь к ним снисходительно. В конце концов, они просто отъявленные разгильдяи, которых лучше всего пересадить в отдел бета-тестирования. Хотя нет – так вы просто заморозите проблему, в итоге она все равно может проявиться. Если разгильдяй действительно любит писать код, при условии, что вы уделяете ему достаточно внимания, он имеет шансы реабилитироваться. Всех, кому это не удается, нужно просто пнуть под зад или познакомить с консультантом по трудоустройству.

Тормоз. Тормоз – это программист, который не знает, с чего начать. Он постоянно ищет спецификацию (или ожидает, пока ему дадут), отчаянно надеясь, что она станет для него отправной точкой. Нерешительность в чем-то хороша, поскольку в некоторых случаях она повышает качество кода. Однако иной раз она свидетельствует о низкой квалификации программиста, который не хочет лишних ошибок на этапе прогона. Предоставьте этим ущербным образец кода, чтобы они могли разобраться, с чего начинать, и выбрать стиль, которого им нужно будет придерживаться. Нерешительность часто характерна для неопытных программистов, и, воспользовавшись некоторыми воспитательными методами, вы можете наставить их на путь истинный. Кроме того, нерешительностью иногда страдают программисты, у которых по тем или иным причинам не слишком впечатляющий послужной список. Ну, скажем, в прошлый раз их результаты разнесли в пух и прах, а теперь они хотят исправиться, но очень боятся наступить на те же грабли. Действительно, низкая самооценка часто проявляется в форме нерешительности. С такими типажами нужно проявлять терпение. Помогите тормозу регулярно добиваться небольших успехов, и тогда все наладится. Наставничество (лучший,

[8] Вы тоже ненавидите пользователей?! Представляете, как было бы здорово писать программы только для программистов?

кстати говоря, способ перевоспитания нерешительного программиста) рассматривается в главе 8. В ней я утверждаю, что роль наставника – это одна из основных ролей руководителя программистов, которая при должных вложениях обязательно окупится.

Любитель. Любители очень хотят стать настоящими программистами. Тщательно изучив какой-нибудь инструмент написания макрокоманд, они возводят себя в ранг хакеров. Единственная причина, по которой они бросают уютные места в отделах поддержки пользователей и тестирования, заключается в том, что, по их мнению, быть программистом – это очень круто. Да, мы, действительно, крутые, но, по большому счету, это лишь побочное следствие нашей основной деятельности. Любителям не хватает образования, но по мере их обучения вы должны пристально за ними следить и лишь при условии определенных достижений с их стороны поручать им работу над критически важными приложениями. Узнав на собственном опыте, как трудно заниматься программированием и какое серьезное внимание к деталям требуется от программистов, любители часто разочаровываются в своем выборе. Они отказываются признавать превосходство объектно-ориентированных методов над процедурной парадигмой – и все потому, что нужное прозрение их еще не посетило. В защиту любителей вспомним замечательное высказывание: «любители построили ковчег, профессионалы построили ”Титаник“». На самом деле иногда свежий, незашоренный взгляд начинающего программиста очень помогает нам – старым брюзгливым технарям.

Профан. Программист-профан – это тот, кого называют тупицей. Хуже всего, когда профан не догадывается о своей тупости. Остерегайтесь таких людей. Иногда они могут достаться вам в наследство от предыдущих руководителей, но сами, я вас прошу, никогда их не нанимайте. У меня нет никаких предубеждений относительно умственно ущербных людей, но я твердо уверен, что в профессии, требующей постоянного самосовершенствования и обучения, таким не место. Если человек невежда, но хочет стать лучше, – дайте ему шанс. Отправьте его, например, в отдел тестирования – иногда не отличающиеся выдающимися умственными способностями пользователи находят себя в отлове жучков[9]. Еще одно соображение насчет глупости: на самом деле все мы постоянно страдаем от несовершенства того, что находится между клавиатурой и стулом. Но, в конце концов, если бы для написания кода не требовались мозги, этим занимались бы все без разбору, так ведь? Я советую не путать невежество с глупостью. Невежество исправимо, а с глупостью лучше просто не связываться. Если вы унаследовали кадры, подобранные не программистом, вполне возможно, что среди ваших подчиненных есть такие типажи. Руководители, имеющие весьма отдаленное представление о технологии, иногда покупаются на необоснованные заверения бездарных претендентов на место.

Эклектик. Можно сказать, что эклектики просто стряпают программные продукты. Представитель этой породы сочетает в себе качества инженера, разгильдяя и не слишком талантливого художника, причем упомянутые ингредиенты находятся в чудовищной диспропорции. Результат их деятельности представляет собой винегрет из стилей кодирования и подключаемых модулей при невероятной путанице в коде. Все это выглядит довольно привлекательно, но стоит лишь попробовать кусочек, как наступят необратимые последствия. Отправьте такого программисты на кулинарные курсы и обязательно проверьте, не скрывается ли за внешней оболочкой талантливости банальный разгильдяй. В классическом виде эта дворянская порода встречается довольно редко, а упомянул я ее по той причине, что отдельные ее черты проявляются в стилях кодирования самых разных типов программистов. Если они не считают нужным следовать корпоративным стандартам, вам придется посвящать все рабочее время напряженным попыткам выяснить, что же они все-таки имели в виду и как сопровождать их код. Основным средством реабилитации эклектиков служит критика кода (см. главу 6).

---

[9] Лично я предпочитаю термину «жучок» (bug) словосочетания «программная аномалия» (program anomaly) и «открытие недокументированной характеристики» (Undocumented Feature Offering, UFO).

Умение обращаться с представителями разных пород

Программисты – это в первую очередь люди. Поэтому в одном человеке могут быть в большей или меньшей степени выражены все перечисленные характеристики. Некоторые из них как будто исключают друг друга, но на самом деле это не так. Все люди сотканы из противоречий, и ваши подчиненные – не исключение. От вас как от человека, осуществляющего руководство этими чудесами природы, требуется понимание, умение мотивировать и, прежде всего, мудрость, которая нарабатывается только с опытом. Мнение о программистах нужно составлять по тем граням их характера, которые ярче других сверкают в свете новых начинаний и ослепляющих вспышек проектов, приближающихся к сдаче.

Мнение о программистах нужно составлять по тем граням их характера, которые ярче других сверкают в свете новых начинаний и ослепляющих вспышек проектов, приближающихся к сдаче.

Предположим, у вас есть счастливая возможность набрать сотрудников в свой отдел с «чистого листа». Какие породы сочетаются удачнее? По-моему, лучше всего соблюдать баланс между архитекторами и конструктивистами. Эти две породы привносят в процесс создания программных продуктов наиболее востребованные навыки – первые мыслят стратегически, вторые прекрасно ориентируются в деталях. К этому альянсу время от времени имеет смысл подключать художников. К сожалению, скорее всего, подобрать группу из идеальных кандидатов не удастся. Работать вам придется с тем, что есть. Потому успех вашего взаимодействия с людьми, сочетающими в себе вышеупомянутые характеристики, зависит от вашей проницательности, терпения и умения быть для подчиненных наставником – то есть от трех универсальных качеств руководителя.

Есть еще один тип личности, на который следует обращать особое внимание. Я имею в виду программистов-ковбоев. Этот тип плохо согласуется с перечисленными породами, а описывать его лучше в соответствии с тем мнением, которое ковбой о себе формирует. Итак, программист-ковбой обычно в совершенстве владеет своим ремеслом, но при этом управлять им практически невозможно. Ковбои глубоко убеждены, что могут работать только над теми проектами, над которыми хотят, делать это на собственных условиях, согласуясь исключительно с собственными планами и обращаясь только к подходящим по их мнению средствам. Такой программист – своеобразный волк-одиночка (или, если придерживаться терминологии этой книги, – кот, который гуляет сам по себе). В зависимости от того, что вам нужно, и вашей готовности терпеть своеобразие их личности, ковбои могут творить либо чудеса, либо хлам. С ковбоями надо держать ухо востро: они ни при каких обстоятельствах не станут частью вашей команды. Прибегать к их услугам стоит либо в безвыходных ситуациях, либо если разрабатываемый проект должен радикально отличаться от всех других, а сопровождать его будут сторонние специалисты.

Почему в программистах сочетаются все эти чрезвычайно занимательные личностные характеристики? Мне кажется, связано это с тем, что сам характер деятельности разработчика программного обеспечения привлекает людей совершенно определенного рода. В своем классическом труде «The Mythical Man-Mouth» Фредерик Брукс (Frederick Brooks)[10] утверждает, что наше ремесло приносит людям удовольствие пяти видов.

1. Радость созидания.
2. Радость созидания полезных для других людей продуктов.

[10] Frederick P. Brooks, The Mythical Man-Month: Essays on Software Engineering, Anniversary Edition (New York: Addison-Wesley, 1995), p. 230. Это действительно непреходящая классика. В нашей области очень мало книг, которые переиздаются через 25 лет после появления, и это – одна из самых стоящих.

3. Привлекательность процесса упорядочивания головоломных объектов, состоящих из взаимосвязанных динамичных элементов.

4. Радость от постоянного обретения новых знаний и решения нестандартных задач.

5. Интерес к работе с продуктами, созданными исключительно путем приложения интеллектуальных усилий человека, которые, тем не менее, существуют, развиваются и делают совершенно непередаваемые вещи.

Все эти факторы кажутся тем людям, которыми вы руководите, чрезвычайно привлекательными. Разобравшись в их мотивации (да и в своей тоже), вы сможете серьезно усилить свои позиции как руководителя.

Кошачьи разборки – соревнование по шипению и царапанию

Проектное совещание превратилось в словесную схватку между Джоном и Кевином. Мы уже перешли к обсуждению регистрации пользователей в системе, а они все еще спорили насчет низкоуровневых подробностей методик конструирования. Дискуссия застопорилась – ведь, несмотря на отсутствие четкого плана обсуждения, все мы были уверены в том, что поговорить есть о чем. Джон и Кевин спорили без перерыва. Дело в том, что Джон (консультант) и Кевин (опытный сотрудник и талантливый программист) руководствовались совершенно разными мотивами и планами относительно этого совещания. Каждый из них считал своим долгом доказать другому, что он умнее, и вопросы проектирования системы их в тот момент не интересовали. А все потому, что Джона назначили руководителем проекта – он занял должность, на которую очень хотел попасть Кевин. Нашего начальника на совещании не было, и ни одного желающего выступить в роли посредника не нашлось.

На следующий день в центре внимания опять оказались Джон и Кевин – они собачились за каждое проектное решение, и все остальные сотрудники отдела предпочитали помалкивать. К концу второго дня мы сумели согласовать значительно меньше вопросов, чем планировались, а то, что нам удалось, оказалось результатом ожесточенной бойни между двумя воинственными котами – Джоном и Кевином. Остальная часть группы чувствовала себя совершенно дезориентированной. Люди начали сомневаться, удастся ли вообще закончить работу над системой. Значительно осложняло ситуацию то обстоятельство, что перед группой поставили задачу сделать систему как можно быстрее, ибо та ее версия, которая в тот момент находилась на рынке, негативно сказывалась на репутации компании.

Этой короткой историей я хотел обозначить трудности, связанные с введением в одну команду консультантов и программистов. Особенно их отношения осложняются в отсутствие начальника. Вы могли бы справедливо поставить под сомнение разумность выбора в качестве руководителя проекта консультанта. Кроме того, как видите, за неимением четкого плана и механизма разрешения противоречий на проектном совещании сотрудники просто убивают время, подрывая тем самым принцип «сначала проектирование – потом кодирование». Развитие межличностных отношений в группе разработчиков подтверждает мою мысль о необходимости психологического анализа подчиненных для повышения качества руководства.

Конец этой истории довольно печален. Проект закрыли, а проектирование и конструирование продукта поручили группе разработчиков из другого отдела. Пропустившему основные «военные действия» руководителю отдела по возвращении пришлось в течение нескольких дней с большими трудностями объяснять начальству, почему после всех обещаний и заверений, предшествовавших началу работу над проектом, разработчики так и не сдвинулись с мертвой точки.

Слава, почет и деньги

Любой сотрудник жаждет признания своей деятельности, хочет ощущать весомость своего вклада в общее дело. Быть оцененными по достоинству стремятся и некомпетентные сотрудники – даже несмотря на то, что их вклад в деятельность компании исключительно негативный. Все мы любим поразглагольствовать насчет того, что создание кода для нас – это уже своего рода награда. Но хотелось бы мне знать, сколько мы проработаем, если нам не будут за это платить. Но даже если будут платить, и платить хорошо, мы в любом случае будем требовать признания со стороны наших товарищей и надеяться, что начальник между делом похлопает нас по плечу. Настоящие кошки предпочитают спокойно дремать в углу, не заботясь о том, кто и как на них смотрит; кошки-программисты в этом отношении, напротив, проявляют некоторый эксгибиционизм. Ярким подтверждением того, что и программистам нужно признание, служит практика встраивания в код своеобразных «пасхальных яиц» (хотя сегодня она считается несколько вульгарной). Вас, начальника, они воспринимают как зрителя, сидящего в переднем ряду, и, конечно же, ожидают аплодисментов. Причем ожидают все – даже те, над которыми впору начинать читать молитву[11].

Расхваливать своих сотрудников – это, конечно, замечательно, но иной раз стоит на секунду остановиться и поразмыслить, отрабатывают ли они те деньги, которые получают. Ведь это входит в вашу задачу как руководителя. Если хотите похвалить человека, сделайте это на виду у всех. Если считаете нужным покритиковать его, не выносите свои оценки на суд публики. Дело не просто в вежливости – эти правила поведения важны в том смысле, что как похвала, так и порицание одного человека на самом деле оказывают грандиозное влияние на всю группу. Поверьте мне, публичное унижение не способствует повышению продуктивности работы группы. Напротив, похвала, высказанная при всех, причем искренне и по заслугам, способна творить чудеса. Не надо кричать о том, что ребята прекрасно поработали, выходя из комнаты для совещаний. Выберите такое время, когда слова благодарности окажут наибольшее воздействие. Четко определитесь с тем, за что вы хвалите человека, и обязательно сделайте так, чтобы сотрудники группы это знали.

Расхваливать своих сотрудников – это, конечно, замечательно, но иной раз стоит на секунду остановиться и поразмыслить, отрабатывают ли они те деньги, которые получают.

И еще немного о поощрении. Кто знает – может быть, у вас у самого такой начальник, который никогда вам слова доброго не скажет. Тогда и вам будет не до комплиментов подчиненным. Роль лидера предполагает стремление к успеху подчиненной вам группы. Когда группа добивается успеха, вы ее хвалите. Кто же хвалит вас? Иногда никто, хотя время от времени вы сами хотите получить от начальника хлопок по плечу. Роль руководителя, которая предусматривает приложение некоторых усилий для улучшения репутации подчиненных, превратится в сущий ад, стоит вам попытаться искать почестей для самого себя. Не стоит забывать, что любой руководитель должен оценивать свои успехи исключительно по тому, насколько эффективно работают его подчиненные.

Если вы выдвинулись в руководители из программистов, задача усложняется, поскольку теперь вам придется принять ответственность за профессиональную судьбу ваших друзей. Но не позволяйте дружбе мешать делам. Лучше используйте ее как средство достижения общей цели. Я вас уверяю, что если вы, в конечном итоге, все вместе добьетесь успеха, ни один из

[11] Я имею в виду тех, кому пора готовиться к основательной порке.

ваших подчиненных не осмелится утверждать, что вы манипулируете дружескими отношениями.

Мотивирование деньгами

Кажется, я уже поднимал денежный вопрос? Лучше было бы его как-то обойти, поскольку, как сказано в Библии, «…ответом всему – монеты»[12]. Согласно последнему статистическому исследованию, почасовая ставка программистов составляет от 30 до 150 долларов, причем большинство из них находится где-то посередине этого спектра. Как определить, стоит ли платить вашим сотрудникам именно столько, сколько вы им платите? Среди факторов, которые нужно учесть, – производительность, опыт, результативность, своевременность исполнения задач, текущая средняя ставка, экономическая конъюнктура и корпоративные стандарты, касающиеся оплаты труда сотрудников в области высоких технологий. Подбирая новых сотрудников и повышая оплату старым, вы должны быть справедливым и бережливым одновременно.

Справедливым и бережливым. Хм, это довольно трудно – можно поддаться соблазну разбрасываться деньгами, как будто они растут на деревьях, опрометчиво полагая, что оплата повышает производительность. На самом деле здесь стоит хорошенько подумать, ведь сегодняшняя роскошь есть завтрашняя потребность. Деньги, подобно власти, оказывают на человека самое что ни на есть деструктивное влияние. И не заставляйте меня цитировать еще одно известное высказывание из Нового Завета[13]. Так все-таки, возвращаясь к теме, как соблюсти баланс в вопросах денежных выплат? Представим себе весы правосудия. На одну чашу положим справедливость, на другую – бережливость. Вес чаши справедливости равен опыту и производительности программиста. Вес чаши бережливости состоит из стандартных коммерческих хитростей, таких как соблюдение баланса доходов и расходов и статистика средней заработной платы программиста. Принимая решение о денежных выплатах, имейте в виду эту аллегорию – она очень действенна.

Но ведь это все теория. А что на практике? Именно из-за несоответствия теории и практики денежный вопрос в деятельности руководителя становится одним из самых сложных. Вам известны принципы, следуя которым вы теоретически должны принимать решения относительно вознаграждения персонала. С другой стороны, существенные коррективы в планирование вносят экономические аспекты – в частности, текущая коммерческая конъюнктура и корпоративная политика. В некоторых компаниях, помимо оклада, сотрудники получают бонусы, которые выписываются в соответствии с личными заслугами каждого. Этот стимул действует лишь тогда, когда сотрудники оказываются в состоянии выполнять то, за что они теоретически заслуживают дополнительного вознаграждения. В принципе, можно ввести практику выплаты ежеквартальных бонусов, но, по моему опыту, это не самый лучший выход – однажды начав их выплачивать, вы обязательно столкнетесь с тем, что сотрудники будут на них надеяться. Касательно денежного вопроса вам лучше проконсультироваться со своим начальником – скорее всего, вместе вы сможете разработать оптимальный для своей команды план. Если решения о вознаграждении принимаете только вы, действуйте так, как считаете нужным, и не забывайте правило справедливости и бережливости.

---

[12] Ну вообще-то в Библии это высказывание приписывается глупцу. Контекст смотрите в Экклезиасте 9:14–19. Только постарайтесь не слишком падать духом, когда будете читать.

[13] См. Новый Завет, 1-е послание к Тимофею 6:10 – любовь, деньги и зло он виртуозно сводит в один силлогизм.

Уровень мышления

[14]

Ну что, вам интересно? Вы заинтригованы тем, что будет дальше? Думаю, что нет. Скорее всего, вы пребываете в полной уверенности, что дальше я разражусь мириадами советов о том, каких действий лучше избегать, а на каких делать упор, и все эти сведения будут представлены в виде маркированных списков. Действительно, в последующих главах таких списков будет немало, но в данный момент я хочу обратить внимание на то, как важно в вашей новой роли думать. Поэтому с перечислением рекомендуемых и не рекомендуемых приемов руководства пока что повременим. Возможно, необходимость думать – это самая сложная обязанность менеджера. Тем не менее это абсолютно необходимое условие нашего успеха. Как пишет в книге «Dynamics of Software Development» Джим Маккарти (Jim McCarthy):

«Основная задача руководителей процесса разработки программных средств состоит в том, чтобы аккумулировать как можно больше интеллектуальных ресурсов и направить их на создание конечного продукта»[15].

Стоя в душе – думайте. Катаясь на велосипеде, прогуливаясь по парку, выделывая невообразимые трюки на роликах – думайте. Сталкиваясь с дилеммами, которые обусловлены принятыми проектными решениями, – думайте. Думать значительно полезнее, чем смотреть телевизор или бесцельно бродить по Сети, – пусть даже там 500 каналов, но на самом деле на них ничего не происходит, и то, что они как будто избавляют человека от необходимости мыслить, совершенно не здорово. Думайте напряженно, до изнеможения, а когда не останется сил – начинайте заново. Результат вас удивит.

Ну а теперь подумаем о том, как справляться с некоторыми типичными ситуациями.

Предположим, в вашем подчинении кот породы минималист, но при этом весьма профессиональный. Вы хотите поручить ему модернизацию продукта, который писал кто-то другой, но доработать который в контексте текущих коммерческих задач совершенно необходимо. Взглянув мельком на код, который вы собираетесь вменить ему в обязанность, он говорит: «Нет, это слишком сложно – код нужно переписать». Естественно, речь идет о коде, который писался два года и который уже сейчас приносит компании неплохие деньги. Ситуация осложняется тем, что человек, который этот код написал, у вас больше не работает и поэтому физически не может помочь новому сотруднику в нем разобраться. Итак, у вас два выхода. Первый: пасть под давлением минималиста, сделав его самым счастливым человеком, но загубив последний шанс сдать проект в срок. Второй: направить его на путь изучения существующей архитектуры и дать ему впоследствии возможность внести в нее весомый вклад. Поиграйте с его пристрастием к четкой архитектуре – попросите документировать существующий код с тем, чтобы в будущем, если позволит время, он смог бы переписать некоторые объекты, сделать их более удобными. Если он профессионал, не сомневайтесь – он обязательно разберется в том, что написал его предшественник. Не стесняйтесь использовать в своих целях дух соревновательности. С точки зрения минималиста все, что написал не он, – полная туфта. На самом деле вполне возможно, что он боится, будто не сможет разобраться в существующем коде, и просто не хочет в этом признаться. Всегда ищите скрытые мотивы, которыми руководствуются все без исключения представители рода человеческого. Поймите: программисты, не готовые признать, что их компетенции не хватит для решения поставленной задачи, очень любят выискивать своему бездействию высокоинтеллектуальные оправдания.

---

[14] Игра слов «шлюзовой уровень» – thunking layer, «уровень мышления» – thinking layer. – *Примеч. перев.*

[15] Jim McCarthy, Dynamics of Software Development (Redmond, WA: Microsoft Press, 1995), p.5.

Как поступить с архитектором, который уверен, что созданные им объектные решения превосходят все созданное ранее, а вы, тем не менее, усматриваете в них некие слабые стороны? Не надо ему ничего говорить про врожденные недостатки решений – ничего не добьетесь, зато наживете себе врага. Лучше попросите его объяснить предполагаемый механизм функционирования всех динамичных элементов и построить несколько прототипов, или тестовых программ, которые смогли бы наглядно продемонстрировать действие заложенных в решении функций. Если он создаст эти прототипы и никаких проблем не возникнет, значит, возможно, вы были неправы, и изъянов в найденном решении нет. Если архитектура кажется вам слишком громоздкой и монолитной, попросите разбить ее на компоненты. Если окажется, что они прекрасно друг с другом работают, значит, архитектор вполне адекватно представляет себе, что он хочет. Если объекты излишне взаимосвязаны и взаимозависимы, это свидетельствует о пристрастии архитектора к усложнению, которое способно существенно удорожить сопровождение продукта. Что отличает высокопрофессионального архитектора? То, что он способен создать конструкцию, которую сможет обслуживать и расширять любой его последователь. Такая конструкция не рассыплется при первой же модернизации кода. На код при работе с архитектором нужно смотреть его глазами – даже если он слеп на один глаз и не видит другим.

> Есть твердое правило: прежде чем пытаться утвердить то или иное решение, используя свое положение руководителя, обязательно выслушайте человека и попробуйте его понять.

Какие еще рекомендации я мог бы дать по поводу такого рода ситуаций межличностного общения? Есть твердое правило: прежде чем пытаться утвердить то или иное решение, используя свое положение руководителя, обязательно выслушайте человека и попробуйте его понять. В том, что касается конфронтации, программисты ничем не отличаются от остальных людей – они хотят, чтобы их выслушали. Как пишет в своей книге «The 7 Habits of Highly Effective People» Стивен Кави (Stephen Covey), «сначала старайтесь понять… и только после этого – быть понятым». Поиск консенсуса при принятии технических решений есть не что иное, как вид искусства, основывающийся на готовности выслушивать чужие идеи. Для того чтобы выстроить такую основу для взаимодействия с сотрудниками, требуется терпение, и проявлять его необходимо – хотя иногда нам кажется, что времени нет даже на конструирование, а насчет методов все вроде бы согласны. Вы можете так считать, но это не отменяет постановку задачи по достижению консенсуса[16]. О том, как достичь консенсуса, мы еще поговорим в главе 5, которая посвящена проведению проектных совещаний. Возможно, вы удивитесь, когда узнаете, что консенсус нельзя строить на основе компромисса.

И еще один пример, иллюстрирующий принцип «услышь, прежде чем судить». Некоторые языки программирования – и, в частности, Visual Basic (VB) – не предусматривают полноценных конструкторов объектов. Недавно я столкнулся с тем, как один художник с помощью события инициализации класса VB пытался организовать обращения к набору объекта со стороны (родительского) класса-потребителя[17]. Если объект VB не удается конкретизировать, перехватить ошибку становится очень трудно. Когда я спросил этого деятеля, почему для обработки подверженной ошибкам операции он выбрал упомянутое событие, в ответ он сообщил, что его решение изящно, понятно и не требует от вызывающего объекта подготовки обращения к интерфейсу. Я, естественно, посчитал, что надежная обработка ошибок значительно важнее любых попыток вылизать текст. Но промолчал. Ознакомившись с его аргумен-

---

[16] Существует мнение, согласно которому полное и всеобщее согласие приводит лишь к тому, что в случае неудачи не остается никого, кого можно было бы в ней обвинить. Может, это и так, но, будучи руководителями, мы не должны увлекаться поиском виновных – значительно полезнее посвятить свое время решению проблем.

[17] Событие инициализации в VB не принимает и не возвращает никаких параметров.

тацией, я объяснил ему, с какими трудностями может столкнуться такой объект, и подкрепил свои соображения наглядным примером в коде. Если бы я сразу сказал что-нибудь вроде «это не есть правильно – разберись!», то не смог бы образумить его своим примером, и он так бы не понял, чего от него хотят. Повторюсь: если дать человеку возможность высказать его точку зрения, он в ответ проявит понимание к вашей позиции.

Как вы адаптируетесь

Из этой главы вы почерпнули множество новых идей. Возможно, вам немного не по себе от масштаба изменений, которые приходятся на долю руководителя на пути самосовершенствования. Не беспокойтесь. В конце концов, на 99 % люди до сих пор генетически идентичны обезьянам, а оставшийся процент сформировался далеко не сразу. Последующие главы, в которых раскрываются другие аспекты руководства и менеджмента, помогут вам адаптироваться, преодолеть трудности и достичь успеха.

Теперь повторим пройденный материал – благо основные принципы руководства должны обосноваться в вашей голове надолго.

• *Вы должны уметь адаптироваться*. Нарабатывая навыки руководства, человек должен приспосабливаться к новому для него социальному контексту. Вы – начальник, а потому ваши взаимоотношения с подчиненными должны серьезно измениться.

• *Самосовершенствование важнее, чем имидж*. Лидерство есть внутреннее качество человека, оно ни в коем случае не обусловливается внешними признаками. Оставаясь программистом, вы как руководитель должны работать над решением сложных проблем, предусматривающих анализ поведения подчиненных (и, конечно же, собственного поведения).

• *Изучайте своих сотрудников*. Станьте исследователем культуры и личностей программистов. Разберитесь, почему ваши подчиненные пишут код именно так, как пишут; подумайте, как использовать их положительные качества и улучшить положение вещей в неблагополучных с точки зрения производительности областях. Пасти котов – значит заставлять их двигаться в одном направлении. Это основная задача руководителя.

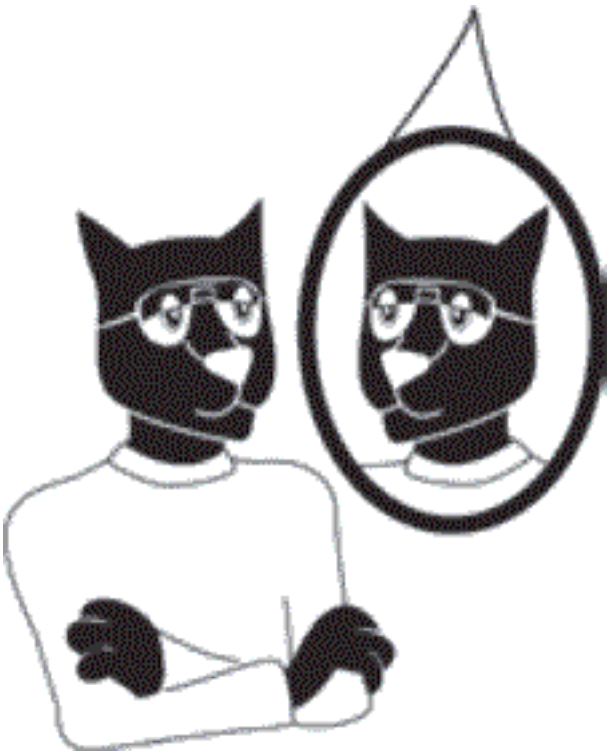
• *Вознаграждайте сотрудников согласно их заслугам*. В вопросах устного и денежного поощрения действуйте по ситуации. Принимая во внимание все финансовые ограничения, старайтесь быть одновременно справедливым и бережливым. Не забывайте, что хвалить подчиненных лучше публично, а ругать индивидуально.

• *Думайте*. Мобилизуйте ваши знания о людях и на их основе старайтесь находить консенсус. Прежде чем судить, выслушивайте аргументацию собеседника. Воспитывайте свой мозг – поскольку вы теперь руководитель, размышления должны стать вашей второй натурой. Готовые варианты решения личностных проблем не способны заменить ваших стараний, направленных на устранение трудностей и развитие возможностей, характерных именно для вашей компании.

Что дальше

В этой главе мы анализировали ваши кадры; в следующей будем изучать вас. Я задам несколько довольно трудных вопросов, призванных выявить ваше отношение к роли руководителя в высшей степени важного процесса конструирования программного обеспечения в контексте текущей конъюнктуры. Чтобы стать хорошим руководителем, вы прежде всего должны научиться управлять собой.

Глава 2. Как руководить собой



Создание в срок качественного программного обеспечения – ваша цель, а управление процессом – ваша работа, так что же вы делаете в свободное время? Вероятно, пишете код. Если вы доросли до начальника из программистов, вам стоит продолжить программировать, в противном случае ваше увлечение этим ремеслом будет ослабевать, и ваши навыки постепенно «сойдут на нет». Другой возможной причиной для того, чтобы искать и находить время для кодирования, является удовольствие, которое оно доставляет. Если возглавлять команду программистов – для вас дело новое и вы пока еще адаптируетесь к этой роли, то кодирование доставит вам удовольствие, поскольку именно это вы хорошо знаете и умеете делать продуктивно. Я полагаю, что продолжать писать код абсолютно необходимо, поскольку это занятие связывает вас с прошлой жизнью, с вашими корнями. Корни очень важны, ведь они определяют ваше отношение к своему ремеслу, и как руководителю вам придется пустить в грунт своей жизни несколько новых корней.

Исследование самого себя даст начало этим корням, так что занимайтесь самоанализом во время ваших ночных бдений над кодом. В предыдущей главе мы рассмотрели принципы взаимодействия с подчиненными, теперь пришло время взглянуть на того, кто ими руководит, – на вас. В этой главе рассматриваются вопросы руководства собой. Каким руководителем я являюсь? Под кого надо подстраиваться, под начальников или под подчиненных? А может, одновременно под тех и других? Ответы на эти вопросы чрезвычайно важны для вашего роста и успеха как руководителя.

Взгляд в зеркало

Исследование вашей объективности может оказаться трудной задачей, поскольку в этом процессе существенную роль играет субъективность. Для того чтобы облегчить самопроверку, взгляните на следующий перечень вопросов и обдумайте их по ходу чтения этой главы. В ее конце я предложу свои ответы на них. Эти ответы я считаю естественными для управленца, и они помогут вам стать лучшим руководителем, чем вы есть. Как теннисист, вы не можете сделать хороший замах, не сжав рукоять ракетки. Управление – это ракетка, так что предложенные вопросы требуют ваших ответов. Итак:

• Считаете ли вы, что предельные сроки завершения проекта – это рекламный прием, на самом деле не имеющий большого значения?

• Позволяете ли вы программистам самим принимать основные архитектурные решения, если вы слишком утомлены или заняты?

• Надеетесь ли вы, что ваши опытные программисты и без вас все сделают правильно, поскольку у вас просто нет времени на то, чтобы нянчиться с ними?

• Полагаете ли вы, что при приближении срока сдачи проекта все проблемы в конце концов решатся[18]?

• Считаете ли вы, что пользователи никогда не сделают ничего, что привело бы к программному сбою, просто потому, что у них не хватит ума для этого?

• Предпочитаете ли вы при управлении коллективом искать консенсус, даже если это требует массы времени и терпения?

• Считаете ли вы электронную почту эффективным средством общения при работе над проектом?

• Согласны ли вы проговорить по телефону несколько часов кряду, только чтобы убедиться, что все идет как надо?

• Считаете ли вы, что с помощью комитетов и комиссий невозможно выработать адекватные бизнес-требования?

• Считаете ли вы себя самым умным программистом в компании?

• Чувствуете ли вы ревность и страх, когда смотрите на действительно хороший код, который написан не вами?

• Делаете ли вы все возможное, чтобы успеть закончить проект в срок[19]?

Ответы, которые вы даете на эти вопросы, могут меняться в зависимости от обстоятельств[20]. Тем не менее некоторые из ответов правильны вне зависимости от текущей ситуации на административном поприще. Надеюсь, что последующие разделы подготовят вас к тому, чтобы отвечать не задумываясь. Возможно, с некоторыми истинами вам будет трудно согласиться, но я полагаю, что, познав самого себя, вы сможете принять их и действовать в соответствии с ними.

---

[18] Это то же самое, что предпочесть синюю пилюлю красной в «Матрице», культовом фильме многих программистов.

[19] Южанин бы сказал: «Я пойду за этим в пасть дьявола». Это значит, что вы ни перед чем не остановитесь ради того, чтобы достичь своих целей. Природа этой аллегории происходит от изображения ада в виде ужасной разинутой пасти, готовой поглотить вас, – очень напоминает срок сдачи проекта, к которому невозможно успеть. Вы проявляете твердость и с мечом в руках рубитесь у врат ада, преодолевая силу, стремящуюся вас уничтожить.

[20] Помните «Звездные войны»? Что Оби-Ван, говоря об отце Люка, сказал о природе правды и точки зрения? (Когда умер Йода.)

Рай, ад, чистилище и ваше место во вселенной

Наиболее существенное усовершенствование, которое вы можете сделать, руководя разработкой программного обеспечения, – усовершенствовать руководителя.

Наиболее существенное усовершенствование, которое вы можете сделать, руководя разработкой программного обеспечения, – усовершенствовать руководителя.

Уильям Блейк (William Blake) писал: «Тот, кто желает, но не действует, распространяет чуму»[21]. Если вы не исправите слабые места в вашем стиле руководства, вы распространите чуму среди своих программистов. Продолжая наш поэтический экскурс (поэзия – близкая родственница программирования), отметим следующие строки:

…в сердце взгляни,
Растет в нем святое древо,
Листья дрожат, как огни,
Питает их радости чрево[22].

Вы должны ухаживать за этим «деревом». Как Вергилий, ведший Одиссея сквозь чистилище[23], я здесь для того, чтобы провести вас к вашему новому месту во вселенной программного обеспечения.

Осознание вашего места в происходящих вокруг вас событиях поможет вам стать лучшим руководителем, чем вы были прежде. Давайте оглядимся в этой новой и прекрасной вселенной.

Ваша работа в корне меняется

Ну что ж, вы не в раю, это уж точно. Наверняка вы осознали это в первый же месяц, распределяя задачи в коллективе. Хотя вы можете подумать, что в раю живет ваш начальник, смею вас уверить, что и это не так. Вы видите, как он планирует текущие задачи и проводит планерки на уровне компании, касающиеся отдаленного будущего, но вы никогда не видели, чтобы он делал что-то, имеющее какую-либо ценность само по себе. Ваше впечатление ошибочно и показывает, насколько вы нуждаетесь в изменении точки зрения на выполняемую работу. Вскоре мы подробнее поговорим об этом, в особенности в главе 9, где наше внимание будет сфокусировано исключительно на отношениях с начальством.

Вы уже поняли, что кодирование больше не ваш хлеб. Иногда вы относитесь к этому как к «программистскому аду», поскольку получаемые вами задания должны быть выполнены к нереальным срокам, и вам не с кем посоветоваться, кроме себя. Как программист вы привыкли к жаре, получали удовольствие от общения с вашими собратьями-программистами и считали программирование основным делом своей жизни.

---

[21] Эту и многие другие прелестные вещи, в равной степени поучительные и гнетущие, можно найти в поэмах Блейка «Marriage of Heaven and Hell». Упомянутая выше цитата взята из «Proverbs of Hell» – William Blake, The Complete Poetry and Prose of William Blake, ed. David Erdman (Derkeley, CA: University of California Press, 1982).

[22] Уильям Батлер Йейтс (William Butler Yeats), избранные поэмы (New–York: Collier Books, 1986). Перевод В. А. Савина (http://zhurnal.lib.ru/s/sawin_w_a/rtfrtf.shtml).

[23] Вы можете освежить ваше классическое образование, прочтя современный перевод Чистилища Данте. Когда вы окажетесь в роли руководителя, вам может показаться, что вы находитесь где-то между раем и адом, однако в действительности это не так – это наше видение обычной жизни.

Теперь у вас появилось другое место – место начальника программистов. Вы где-то посередине между раем и адом и поэтому наслаждаетесь преимуществами обоих. В мифологии такое промежуточное место между раем и адом называется чистилищем. Именно в нем устраняются последние преграды на пути в рай. В чистилище грешник также получает свою долю мук, но не таких ужасных, как в аду. На первых порах вы можете не почувствовать, что в этом новом месте страдаете меньше, но когда вы однажды к нему привыкнете, оно вам покажется совсем неплохим. На самом деле я не знаю лучшего места: вы по-прежнему можете писать код, но у вас также есть возможность помочь другим делать ту же работу.

Вам нужно заново учиться оценивать свои успехи, увлечения, амбиции

Геометрические аспекты рая, чистилища и ада вполне применимы к вашему месту в иерархии управления компанией. Представьте себе, что с каждой ступенькой вверх по административной лестнице растет и высота, с которой вам, возможно, придется падать. Если ваша лестница правая (прислонена к правой стороне стенки), мужайтесь: в конце концов, вы знаете, что ваше дело правое. Кстати говоря, а куда вы движетесь как программист-начальник? Будем надеяться, по направлению к успеху – как личному (персональному), так и общему (корпоративному). Хотя успех и определяется по-разному, одно из определений я нахожу наиболее полезным и практичным: это способность радоваться своему труду и не терять своего увлечения. Может показаться, что при такой оценке увлечение ставится слишком высоко, однако оно может зажечь огонь у вас внутри и помочь вам покорить столь непредсказуемый мир разработки программного обеспечения. Увлечение – это топливо, которое может раскрутить «мотор» вашего руководства. Требуйте от своих программистов во всем добиваться безупречности, чего бы это ни стоило. Если они считают, что изящество и безупречность – это те две вещи, которые затягивают время кодирования, напомните им, что элегантность – это достижимое совершенство. Увлечение побудит их добиться этих целей. Лелейте свое увлечение – ведь для вас это единственный способ расти, приспосабливаться, превозмогать, достигать цели. Это часть обязанностей по уходу за деревом, на котором распускаются цветы успеха.

> Лелейте свое увлечение – ведь для вас это единственный способ расти, приспосабливаться, превозмогать и достигать цели. Это часть обязанностей по уходу за деревом, на котором распускаются цветы успеха.

Учитесь обживать свое место, а не рваться вперед. Амбиции могут быть чрезвычайно разрушительной силой, если они отрывают вас от реальности и решаемых задач. Быть амбициозным – значит преуспевать в вашей новой роли руководителя и отчасти программиста. Если в будущем вам выпадет продвинуться по служебной лестнице, вы должны быть уверены, что продвижение по службе – совсем не то, к чему вы активно стремитесь, а просто лучший способ реализовать свои амбиции. Вы сможете делать эту амбициозную работу, если у вас сердце программиста и при этом вы сумели развить в себе мышление руководителя.

> Вы сможете делать эту амбициозную работу, если у вас сердце программиста и при этом вы сумели развить в себе мышление руководителя.

Естественный отбор и время

Из наблюдений за миром живой природы мы знаем, что для естественного отбора, искореняющего недостатки и повышающего выживаемость, время – один из необходимых ингредиентов. В мире программного обеспечения у вас нет такой роскоши, как тысячелетия. Потребителям программного обеспечения может показаться, что на создание новой версии уходит бездна времени, но они ведь просто не знакомы с процессом. Вы разбираетесь в процессе и должны учитывать время в повседневной деятельности. В революционной книге, посвященной вопросам управления разработкой программного обеспечения, пустая трата времени персоналом упоминается как величайший грех управленца[24]. А как вы тратите ваше собственное время руководителя?

Время – это либо союзник, либо враг. Взвесьте, что сказал Френсис Бэкон (Francis Bacon) – один из отцов-основателей современной науки:

«Тот, кто не ищет новых лекарств, должен ожидать появления новых бед, ведь время – величайший новатор»[25].

Немного помогите процессу естественного отбора: продумывайте административные вопросы, которые грозят обернуться потерей времени, вносите изменения и работайте на опережение, иначе вы рискуете оказаться в их власти. В этом разделе я опишу несколько проблем подобного рода. Я не стану останавливаться на деталях управления, они будут обсуждаться в следующих главах. На данный момент я хочу только, чтобы вы поняли: сознательно или нет, но некоторые методики вы уже применяете. Время, растрачиваемое впустую, должно быть вашей постоянной головной болью.

Избегайте ненужных, неэффективных совещаний

Как менеджер и руководитель вы будете участвовать в куда большем количестве встреч, чем в бытность свою программистом. Глава 5 целиком посвящена этому предмету. Давайте обсудим сейчас, какие из этих встреч действительно необходимы, и постараемся сделать их эффективными.

Для общения вашего коллектива вполне достаточно одной встречи в неделю. Остерегайтесь тратить чересчур много времени на разговоры и мало на решения и действия. Не устраивайте встречу только ради того, чтобы получить одобрение своих решений. Поощряйте дискуссию, но ищите решения. И помните, что дьявол – в деталях, а цель любой встречи – это изгнание этих дьяволов. Если вы контролируете ход встречи, вы контролируете время. Установите предел для любой встречи: 45 минут общения обычно более чем достаточно. Зачастую могут быть нужны и 8-часовые обсуждения проекта, однако всегда имейте подробную повестку дня и следуйте ей, если вы собираетесь выдержать такую длительную мозговую атаку.

Не устраивайте встречу только ради того, чтобы получить одобрение ваших решений. Поощряйте дискуссию, но ищите решения.

[24] Tom DeMarco and Timothy Lister, Peopleware: Productive Projects and Teams, Second Edition (New York: Dorset House Publishing, 1999).

[25] См. эссе Бэкона «Of Innovations» (1625). Еще одна занятная цитата: «Время – это огонь, в котором мы горим». Это, возможно, взято из произведения какого-нибудь великого классического писателя, но черт меня возьми, если я смогу выяснить, кого именно, так что нам пришлось ограничиться Бэконом.

Не планируйте слишком мало или слишком много

Не верьте теории, утверждающей, что чистый стол – признак скудоумия. При взгляде на рисунки Эйнштейна может показаться, что на столе у него царил ужасный беспорядок, но я ручаюсь, что в его голове – нет. Вам необходимо в достаточной мере организовать процесс, но не тратить все время на планирование. Достичь равновесия трудно, но необходимо. Тема организации, необходимой для успеха, затронута глубже в главе 4. Не путайте вопросы организации работы с вопросами ее выполнения. Организация – это начальная стадия плана. Разрабатывайте план. Вспомните, что сказал Спок (Spock): «Логика – начало мудрости»[26]. Аналогично, планирование – начало выполнения.

Бессмысленно ожидать чего-либо при отсутствии контроля

Вы роздали задания, вернулись к вашим обязанностям и надеетесь, что все заняты выполнением заданий. Однако после ваших подробнейших объяснений Джо спокойно отправляется бороздить Интернет, даже не попытавшись заставить программный интерфейс приложения работать правильно. Если вы думаете, что промежуточные этапы не являются неотъемлемой частью поставленной вами задачи, то вы понятия не имеете о том, как работает голова программиста, даже несмотря на то, что вы – один из них. Это вполне в человеческой природе: если до срока сдачи остается месяц, времени на то, чтобы сделать работу, вполне достаточно. Время можно считать потерянным, если нет очевидного продвижения. Работа над частью нового продукта – это поступательный итерационный процесс, не подчиняющийся булевой логике и предполагающий много возможностей для проверки. Предположим, вы сообщили отделу тестирования, что дата завершения кода – XX-е января. Лучше, если эта дата не будет рабочей датой окончания работы над кодом! Вы не сможете сделать работу X, если сначала не будет выполнена часть X – Y, где Y – еженедельно определяемая часть работ. (Вы можете сказать, что «сделано за день»?) Вы наверняка слышали, что цена свободы – это постоянная бдительность, ценой же вовремя сделанного программного обеспечения является неизменное усердие.

> Вы наверняка слышали, что цена свободы – это постоянная бдительность, ценой же вовремя сделанного программного обеспечения является неизменное усердие.

Проектируйте архитектуру, прежде чем выбирать технологию

Технология волшебной пули или золотого молотка (как бы вы ее ни назвали) не может решить бизнес-проблем, это делают люди. Уверен, вы применяете технологию для реализации решений, но вы тратите время попусту, если думаете, что покупка последнего дополнения к среде разработки даст скачок производительности. Следующая версия языка программирования также не решит ваших проблем. Конкурирующие производители средств разработки программного обеспечения обещают многое. Наша отрасль разделена надвое: Microsoft и все остальные. Я, конечно, понимаю, что это слишком упрощенное разделение, но оно послужит иллюстрацией моего утверждения: продукты Microsoft могут оказаться не оптимальными для решения ваших конкретных задач, так как Microsoft слишком большая и многоплановая корпорация. Не важно, много или мало у Microsoft возможностей влиять на всю отрасль и каковы эти возможности, – технология Microsoft построена на базе заранее определенного архитек-

[26] Помните как в «Звездном пути» Спок напился с Валарисом после их беседы об изгнании людей из рая? Гм, возможно, рай – это просто работа программиста в сравнении с работой руководителя, – вам решать.

турного плана. Мир Java, олицетворяемый Sun, слегка отличается, и хотя он более фрагментирован по сравнению с Microsoft, Sun также создает свои продукты на основе конкретной архитектурной схемы. Вы можете использовать Enterprise JavaBeans или. NET сколько душе угодно, но в любом случае вам придется принять внутренние архитектурные ограничения. Я призываю определить ваши архитектурные задачи и планы до того, как вы выберете технологию реализации. Вам придется все переделать, если с новым инструментом дело «не выгорит». Вы уже много раз слышали: если у вас не хватает времени даже на то, чтобы сделать все правильно, где же вы его найдете на то, чтобы все переделать?

Баланс между чистотой и практичностью

Поиск равновесия между чистотой и практичностью – трудное дело для человека, влюбленного в программирование. Концепция «хорошего программного обеспечения нужно в меру», возможно, впервые открытая Microsoft, имеет свои достоинства. Вы можете тратить очень много времени, добиваясь чистоты кода в ущерб практичности. Что действительно нужно, так это удобство эксплуатации программного обеспечения, и практичность больше соответствует этой цели, чем чистота. Конечно, чистота может быть полезной, но какой ценой? Я не говорю о создании неряшливого или непродуманного кода, я имею в виду программное обеспечение, которое может дополняться и расширяться не только его создателем, но и другими программистами. Проблема чистоты кода состоит в том, что она подобна красоте – ее воспринимают глазами. Вашим глазам постоянно нужно носить очки практичности.

Не выполняйте задания, а распределяйте их

Классическая управленческая ловушка для начальника, вышедшего из программистов, касается распределения заданий. Вы понимаете, каким образом реализовать определенное решение, а обучение членов команды, которые вовсе не обязательно видят это решение, требует времени. Здесь применима старая китайская поговорка: «Дайте человеку рыбу, и он будет сыт один день; научите его ловить рыбу, и он будет сыт всю жизнь». Если бы все стали столь же сообразительны, как вы, было бы вам проще работать? Возможно. Потратьте время на обучение сейчас, и вы сэкономите его потом, поскольку ваши сотрудники научатся решать проблемы без вашей помощи. Тогда вы сможете эффективно распределять задания, а не давать объяснения.

Документируйте то, что вы делаете или планируете делать

Трудным заданием для тех, кто любит программировать или управлять по наитию, является документирование. При необходимости можно «опередить события» и сделать экранный прототип, но экранные снимки делаются только ради конкретизации проектной документации. Не отдавайте прототип напрямую программисту, который должен завершать проект. Может показаться, что такой прототип сэкономит время, но он может просто повести программиста неверным путем и спровоцировать фальстарт. Не думайте, что для написания кода бизнес-требований достаточно. Бизнес-требования – только начальный этап разработки документа, позволяющий обрисовать архитектуру, а дальше вы можете говорить о реализации решения в объектах кода. У вас всегда будет искушение, когда поджимает время (а разве бывает иначе?), слепить все «по-быстрому», однако надо быть настоящим счастливчиком, чтобы, не имея четкого плана, создать программный продукт, который можно будет в дальнейшем дополнять и обновлять.

Вы можете время от времени ощущать себя во власти блок-схем и диаграмм, но это лучше, чем код, хранящийся в корпоративной базе данных без какого-либо указания на то, как он работает и какую проблему он призван решать. Документирование радикально отличается от программирования, но это необходимый первый шаг на пути превращения идей в продукты. Следующим шагом может быть прототип, но смотрите на прототип как на продолжение документа, а не как на начало проекта. Прототипы служат для обоснования готовой концепции, в то время как программные продукты представляют собой реализацию готового проекта.

Прототипы служат для обоснования готовой концепции, в то время как программные продукты представляют собой реализацию готового проекта.

Оценка вашей производительности

Как я упомянул в этой главе ранее, в момент, когда вы впервые превращаетесь из программиста с полной занятостью в начальника с полной занятостью и одновременно программиста с частичной занятостью, административная деятельность кажется не столь продуктивной, как кодирование. Учитесь различать просто трату нервной энергии и настоящее творчество. Что я имею в виду? Если вы привыкли кодировать часами, то вы поймете, что такое нервная энергия. Это то состояние вашего ума, когда вы прикончили уже три чашки кофе[27], вам еще писать и писать, но вы уже чувствуете приближение результата. Настоящее творчество – это ощущение, когда вы видите и конечный результат, и все ступени его достижения, но при этом знаете, что должное качество будет достигнуто далеко не так быстро. Я знаю, что все сказанное трудно переварить, но вам действительно нужно как следует обдумать эту концепцию.

Как программист вы привыкли измерять свою производительность числом работоспособных объектов, созданных вами в течение дня. Подобный метод оценки может стать причиной вашего провала как руководителя. Вы будете разочарованы и не удовлетворены тем, как вам приходится проводить время, пока не примете новый образ мышления. Как начальнику вам следует оценивать свою производительность объемом работы, которую выполняет ваш коллектив. Вы не можете оценивать этот объем каждый день, и это может стать серьезной проблемой, если вам требуются постоянные подтверждения того, что вы работаете хорошо. Как ежедневно решать эту проблему? Заходите к своим программистам каждый день или звоните им и проверяйте, как идут дела. Когда они привыкнут ждать этих ежедневных встреч, произойдут две вещи: они никогда не будут знать, когда вы появитесь, и, таким образом, постараются быть в «постоянной готовности», а вы сможете оценить их ежедневный прогресс и соответствующим образом подстроить под него свой график работы. При этом обе стороны прекрасно понимают, что происходит, однако приятно притворяться, что никто ничего не замечает.

> Как начальнику вам следует оценивать свою производительность объемом работы, которую выполняет ваш коллектив.

В своей крайне проницательной книге «The End of Patience» Дэвид Шенк (David Shenk) пишет:

«Что касается гипертекста, окончания излишни, поскольку никто никогда их не достигнет. Чтение открывает дорогу к катанию на волнах информационного океана, извилистому, бесконечному странствию через лабиринт тем. Странник создает собственную повесть, выбирая наиболее привлекательную ссылку, которая всегда доступна. Как техника поиска – это роскошно. Однако как способ мышления это имеет серьезные пороки»[28].

По аналогии с приведенной цитатой, не оценивайте вашу производительность тем, насколько быстро вы можете перепрыгнуть от одного проекта (или сотрудника) к другому. Не теряйте из виду масштабные цели и оценивайте небольшие шаги, необходимые для успешного достижения цели. Не занимайтесь самообманом – пусть реальное положение дел диктуется фактами, а не вашим оптимистичным воображением. Ключевые слова из предшествующей цитаты – «привлекательный» и «мышление». Не соблазняйтесь непосредственным удовольствием от кодирования; лучше поразмыслите, как помочь остальным получать это удовольствие под вашим руководством.

---

[27] Подставьте сюда любую любимую вами жидкость, содержащую кофеин. Я предпочитаю кофе программистской крепости, светонепроницаемый и способный почти стоять на столе без всякой чашки.

[28] David Shenk, The End of Patience: Cautionary Notes on the Information Revolution (Bloomington, IN: Indiana University Press).

Как я уже упоминал, управление программистами и разработка программного обеспечения требуют серьезных размышлений. Добавьте к этому перечню настойчивость, качество, которое отнюдь не всегда присуще душе программиста, но которое в случае крайней необходимости может быть привито ей программистом-начальником. Примените часть этой настойчивости к самому себе – и вы обнаружите, что она нужна как программисту, так и его начальнику. Системность – внутренняя дисциплина, привнесенная вами в работу, – помогает быть настойчивым.

Если вы новичок в деле управления, не ожидайте, что почувствуете себя комфортно ранее, чем через 6 месяцев. Пусть этот дискомфорт напоминает вам о необходимости многому научиться, а также о том, что изучение новых способов быть полезным и чувствовать себя полезным требует времени, но за это воздастся в будущем.

Контролируйте свои слабости

Вы великий программист, правда? Я уверен в том, что так оно и есть, а также в том, что вы все же еще не совершенны. Остерегайтесь, что ваши слабости как программиста заставят вас смотреть сквозь пальцы на тех в вашем коллективе, кто похож на вас. Если вы сами не любили документировать и проектировать, прежде чем начать кодировать, тогда вы, вероятно, можете позволить другим этого не делать.

> Если вы не любите документировать и проектировать, прежде чем начать кодировать, тогда вы, вероятно, можете позволить другим этого не делать.

Такие вещи вряд ли могут быть полезными. Если вы минималист, то в случае когда дело доходит до обработки ошибок (которых в идеале не бывает), вы можете не заметить отсутствия процедур обработки ошибок в коде Салли, считающей все настолько очевидным, что обработка ошибок – это просто трата места.

Я могу продолжить перечень слабостей программистов, но оставлю это вам. Тем не менее вам требуется осознать необходимость постоянно быть внимательным к своим недостаткам как программиста, поскольку нельзя прощать другим то, что вы могли бы простить себе. Это может выглядеть противоречиво, и так оно на самом деле и есть, так что продолжайте работу над исправлением своих недостатков как программиста, чтобы не заразить ваших подчиненных теми же слабостями.

Чтобы вам было легче преодолевать свои слабости, представьте себе, что многие люди прошли тот же путь, что и вы. Некоторые из них проложили на этом пути вехи для вас – это книги и иногда URL-адреса. Другими словами, для того чтобы заполнить пробелы в знаниях и/или опыте, рекомендую побольше читать. Вам нужно читать по многим причинам. Вот некоторые из них.

• *Читайте, чтобы оставаться «на уровне»*. Сюда можно отнести чтение отраслевых журналов, книг, посвященных вашему основному языку программирования, и журналов, касающихся методов, применяемых в вашей работе. Это вы должны делать почти каждый день, просто чтобы не потерять необходимые навыки.

• *Читайте, чтобы быть в курсе*. Такое чтение расширит ваши познания, причем некоторые из источников, используемых для сохранения вашей квалификации (чтобы оставаться «на уровне»), применимы и в этом случае. Здесь вам надо сфокусироваться на том, что, возможно, не требуется прямо сейчас, но может понадобиться в ближайшем будущем. Интернет-рассылки, форумы и прочие чудеса информационной эры могут быть необычайно полезными. Не забывайте, однако, что эра информации – это совсем не то же самое, что эра знаний.

• *Читайте, чтобы углубить свои знания*. Подобное чтение может быть сложным, но вы должны находить для него время, чтобы достичь более полных знаний по технологии и методикам, с которыми сталкиваетесь в своей повседневной работе. Здесь вам придется внимательно изучить книжные шкафы в вашей местной библиотеке, книжном магазине или Интернете в поисках книг, которые действительно могут помочь. Для того чтобы углублять свои знания, в выборе книг избегайте характерного для поваренной книги подхода к изложению. Вам нужно нечто большее, чем просто набор рецептов; вам нужно «откусывать больше того, что вы, как вам кажется, можете прожевать».

• *Читайте, чтобы стать мудрее*. Умные люди встречаются не только в нашей области. Другие научные дисциплины, а также художественная литература могут научить вас подходить к любому вопросу с разных сторон. Очень часто в нашей профессии мы ограничиваемся вертикальным мышлением. Вертикальное мышление подразумевает поступательное движение от одного логического умозаключения к следующему и отбрасывание того, что не укладывается в

рамки заранее имеющихся у нас представлений о наиболее подходящем. Разностороннее мышление – это то, что приносит оригинальные идеи и зачастую является признаком гениальности.

Хочу специально сообщить приверженцам технологии: бизнесмены – тоже умные люди. Под бизнесменами я подразумеваю тех, кто многие годы участвует в строительстве крупных корпораций и правительственных органов. Должен признать, что мне далеко не сразу удалось научиться уважать навыки, необходимые для ведения бизнеса. Я был интеллектуальным снобом, считавшим, что если кто-то не понимает всех деталей технологии, то он по определению не может принадлежать к когорте ярчайших и лучших. Это было проявлением слабости. Вы можете многому научиться у руководителей в различных областях, не только связанных с технологией. Расширьте ваши интеллектуальные поиски, включив в них области деятельности, в большей степени связанные с применением не технологии, а деловой хватки. Концентрировать усилия людей на решении бизнес-проблем – ваша основная работа как начальника, а технология – это только одно из возможных решений. Вам нужно ознакомиться с другими путями решения проблемы, так что читайте книги, в том числе и не относящиеся к области технологии.

Вот пример того, что я имел в виду, говоря о деловом подходе. Джек Уэлч (Jack Welch), бывший долгое время главой General Electric, пишет:

«Я полагаю, что бизнес во многом похож на ресторан мирового класса. Если вы заглянете в двери кухни, пища никогда не будет выглядеть так же хорошо, как на вашем столе, куда она попадает великолепно украшенной, в красивой посуде. Бизнес беспорядочен и хаотичен. Я надеюсь, что вы сможете найти на нашей кухне что-нибудь, что могло бы быть полезным в достижении вашей мечты»[29].

Не правда ли, сказано будто про бизнес в области программного обеспечения? Уэлчу есть много чего сказать, причем вполне уместного и для руководителя программистов, поскольку он пишет о руководстве в течение 20 лет одной из крупнейших и доходных компаний. Учитесь у тех, кто проделал путь наверх до вас. Идите по их следам.

Многочисленные ссылки на литературу, встречающиеся на страницах этой книги, призваны помочь вам лучше ориентироваться. Выберите полезные для вас издания и прочитайте их. Книги нужны не только для того, чтобы украшать полки или произвести впечатление на ваших друзей. Хорошие книги – это жизненные откровения. Научитесь определять, кто из авторов обращается непосредственно к вам, и слушайте то, что он пытается до вас донести.

[29] Jack Welch, Straight from the Gut (New-York: Warner Business Books, 2001), p. XV.

Ответы

Вопросы, с которых начиналась эта глава, были задуманы как трамплин для самоанализа. Теперь, когда вы, оттолкнувшись двумя ногами, прыгнули в глубину своей души, давайте познакомимся с ответами на эти вопросы.

• Считаете ли вы, что предельные сроки завершения проекта – это рекламный прием, на самом деле не имеющий большого значения?

На самом деле сроки всегда имеют значение. Ваша компания может распространять программное обеспечение с целью продажи предоставляемых им услуг, и информация о товаре должна содержать описание его свойств. Даже если пользователи не задействуют все возможности вашего продукта, они должны быть представлены в рекламном буклете. Некоторым состоятельным клиентам может потребоваться лишь одна деталь, которая не нужна остальным, и ее поддержка программным обеспечением может стать причиной ощутимой выгоды. Маркетинг может оказывать опасное влияние на разработку программного обеспечения, однако он необходим. Даже если вы не хотите, чтобы весь цикл разработки определялся отделом продаж, все равно для вас он будет одной из движущих сил в определении даты окончания работы над проектом. Постарайтесь узнать продавцов поближе, стать их другом. Заслужите их доверие пунктуальностью и узнайте от них положение вашего продукта на рынке. (См. далее врезку «Кошачьи разборки», непосредственно связанную с этим вопросом и ответом на него.)

• Позволяете ли вы программистам самим принимать основные архитектурные решения, если вы слишком утомлены или заняты?

Вам, скорее всего, придется перепоручать принятие некоторого количества непростых решений, особенно в первые дни вашей карьеры руководителя. Это может сработать, особенно если вы руководите хорошими людьми, однако помните, что возможность перепоручить принятие решений есть у вас не потому, что вы устали, а потому, что лучшие решения могут рождаться не только в вашей голове. Отказ от принятия решения – тоже решение. Пассивности на самом деле не существует, а если она имеет место, то скоро может потребоваться кто-то еще, кто бы мог делать вашу работу.

• Надеетесь ли вы, что ваши опытные программисты и без вас все сделают правильно, поскольку у вас просто нет времени на то, чтобы нянчиться с ними?

Этот вопрос тесно связан с предыдущим – и опять же: если вы руководите квалифицированными людьми, то им может не требоваться ваше постоянное внимание. Только не забудьте о том, что я говорил относительно ожидания результатов без проведения проверок.

• Полагаете ли вы, что при приближении срока сдачи проекта все проблемы в конце концов решатся?

Назовите это принятием желаемого за действительное или, на языке детей, ожиданием чуда. Как вы это ни назовете, полагаться на чудо опасно, и такая надежда обычно является результатом стресса. Многие видят в стрессе обычное жизненное явление, не всегда приводящее к пагубным эффектам. Стресс может быть мощным фактором, направляющим ваши усилия, если вы привыкли постоянно быть под напряжением и не сдаваться. Если вас не заводит ваша работа, возможно, вы выбрали не ту работу. Не бойтесь неудач, они когда-нибудь обязательно случаются; не бойтесь не успеть к сроку, это тоже когда-нибудь произойдет. Учитесь на своих ошибках; не научившись стойко переносить неудачи, вы никогда не будете знать, что делать с успехами.

• Считаете ли вы, что пользователи никогда не сделают ничего, что привело бы к программному сбою, просто потому, что у них не хватит ума для этого?

Это тоже вопрос принятия желаемого за действительное. Из программистов получаются плохие бета-тестеры – поскольку мы подсознательно знаем, что определенные функции при

проверке «грохнутся», мы и не пытаемся их проверять. В конечном счете ошибки всегда проявляются, так что не позволяйте поспешности влиять на качество. Как руководителю проекта вам стоит самому включиться в рабочий процесс на этапе альфа-тестирования, дабы выявить небрежности в программировании.

• Предпочитаете ли вы при управлении коллективом искать консенсус, даже если это требует массы времени и терпения?

Я коснулся этого вопроса в конце предыдущей главы, а в этой подробно осветил роль времени и необходимость сохранять терпение, поэтому полагаю, что ответ очевиден. Если ваша интуиция вас иногда подводит, скажу прямо: консенсус должен быть вашей постоянной целью, и вы должны делать все, чтобы его достичь. Более подробное обсуждение этого вопроса вы можете найти в главах 5 и 6.

• Считаете ли вы электронную почту эффективным средством общения при работе над проектом?

Ну конечно, она таковым не является, но если ваш коллектив пространственно разделен, она иногда остается единственно доступным средством общения. Совместное редактирование документа куда проще, чем запутанная переписка по электронной почте, однако электронная почта необычайно удобна, поэтому люди в первую очередь стремятся использовать именно ее. Только убедитесь, что у вас не накапливается непрочитанных сообщений, – лучше всего создать из них один документ, содержащий все, что имеет отношение к разработке.

• Согласны ли вы проговорить по телефону несколько часов кряду, только чтобы убедиться, что все идет как надо?

Через это надо пройти. В вашей работе телефон – необходимое зло, если только все, с кем вы работаете, не находятся от вас дальше предела слышимости. В некоторых случаях телефон – ваше единственное средство проведения ежедневных проверок. Программы общения в режиме реального времени, наподобие ICQ, иногда заменяют телефон, но отнимают слишком много времени.

• Считаете ли вы, что с помощью комитетов невозможно выработать адекватные бизнес-требования?

Иногда комитеты и комиссии бесполезны, но обычно в большой и сложной организации они необходимы. Теперь, когда вы стали ответственным лицом, вам, возможно, предстоит быть членом или председателем большего числа разнообразных комитетов и комиссий, так что будет лучше, если вы научитесь их использовать. При удачном стечении обстоятельств комиссии могут быть мощным средством объединения интеллектуальных ресурсов, так что не надо недооценивать их значение. Как-никак, ни один человек не обладает всей необходимой информацией, а без знания бизнес-требований ваше программное обеспечение сгодится разве что для забавы.

• Считаете ли вы себя самым умным программистом в компании?

Возможно, вы ответили на этот вопрос положительно. Причиной его задать было желание помочь вам побороть свое самомнение. Всегда, когда это возможно, старайтесь окружать себя теми, кто умнее. Никогда не обманывайте себя надеждой, что вы единственный, у кого есть все ответы. В конце концов, любое, даже самое оригинальное мышление – это просто химические процессы в мозгу, и чьи-то молекулы всегда могут быть лучше ваших.

• Чувствуете ли вы ревность и страх, когда смотрите на действительно хороший код, который написан не вами?

Это вопрос-ловушка. Если вы гордитесь тем, что делаете, то вполне нормально чувствовать легкую ревность при виде чего-то, что сделано кем-то лучше. Под маской этой ревности обычно скрывается страх, что вас перехитрили или, что хуже, кто-то достиг чего-то, чего вы даже не в силах понять. Этот вид внутреннего эмоционального состояния «невовлеченности» часто поражает целые отделы. Осознайте эти ощущения и их причину и двигайтесь дальше.

Ваша работа – качественно и в срок завершить проект, поэтому используйте все, что может помочь вам в этом.

• Делаете ли вы все возможное, чтобы успеть закончить проект в срок?

Это возвращение к первому вопросу. Несоблюдение сроков может нанести убытки вашей компании и похоронить вашу карьеру. Нужны ли здесь еще какие-нибудь слова? Да. Лейтмотивом хорошего руководства разработкой программного обеспечения является прилежание, бдительность, внимание к деталям. Определение крайнего срока лежит в рамках этих составляющих руководства. Вы, может быть, вспомните, как Скотти из «Звездного пути» заслужил свою репутацию уникального работника[30], и примените схожий метод. Только не умножайте вашу исходную оценку времени, необходимого для завершения работ, на произвольное число, а назовите вашим людям какую-нибудь вымышленную дату, предшествующую дате, которую вы сообщите отделу тестирования. Чем больше вы будете практиковаться в оценке сроков, тем больших успехов вы достигнете в этом виде искусства.

Эти вопросы не были тестом – это были размышления о вашем положении начальника, позволившие выявить слабости, которые могли бы помешать вашему успеху.

Кошачьи разборки – похоронный марш

Несколькими днями ранее этого прекрасного весеннего майского дня прошел последний срок сдачи, а работа над кодом еще не была завершена. Пение птиц и свежесть чистого неба напомнили Роджеру о прекрасном, он даже на секунду забыл о проблемах программного обеспечения встроенного процессора.

Поставив машину на свое вице-президентское место парковки в недавно построенном здании штаб-квартиры корпорации, Роджер обратил внимание, что на парковке не было машины Джеффа. Ну что же, еще один разговор о своевременности этим утром будет весьма кстати. Эти разговоры между Джеффом и Роджером за последние несколько месяцев случались часто. Они оба отдавали себе полный отчет в важности завершения нового программного обеспечения в срок. Казалось, что Джефф всегда будет отвечать, что он сожалеет и приложит максимум усилий, для того чтобы система заработала на этой неделе. Вице-президент по продажам уже подписал со многими компаниями по всей стране контракты на установку новой системы контроля и управления, которую Джефф и Роджер должны были наконец-то завершить. Президент дал ясно понять, что поскольку несколько крайних сроков уже прошли, работа обязательно должна быть закончена к этому новому крайнему сроку.

Роджер размышлял о том, что он мог бы сказать Джеффу такого, что было бы внове для него и могло бы заставить его наконец-то завершить работу. Он знал, что Джефф подолгу задерживается на работе вечерами и в выходные, – он звонил ему на всякий случай. Хоть бы код был написан не на C, тогда, может быть, Роджер смог бы помочь. Ну что ж, думал Роджер, он сделал все, что мог.

Прошло около тридцати минут рабочего дня, когда президент заглянул в просторный кабинет Роджера и сказал: «Спуститесь в мой кабинет на несколько минут. Вы мне нужны для небольшого разговора». Роджер поежился и подумал, что «небольшой разговор» – это обычная словесная выволочка от вышестоящего начальства. Когда Роджер присел напротив письменного стола руководителя компании, секретарь закрыла дверь в кабинет. Президент посмотрел в глаза Роджеру и произнес: «Вы догадываетесь, зачем я вас позвал?» Роджер не имел ни малейшего понятия, но пробормотал несколько слов о том, что сожалеет. Президент, как казалось, был в затруднении и просто сказал: «Собирайте ваши вещи. Вы уволены. Я вам месяц назад

[30] Скотти просто умножал оценочное время ремонта на 4.

сказал, что к этому последнему сроку мы должны закончить. Чек с вашим выходным пособием получите у моего секретаря. Я хочу, чтобы вы покинули здание до полудня. До свидания».

Чему вы можете научиться из этой истории? Многому – в ней на каждом шагу намеки. Речь идет об очевидно успешной и богатой компании, отделом продаж в которой руководит очень напористый вице-президент. Он уже подписал контракты на систему, которая даже не была закончена. Другой же неудачливый вице-президент, ответственный за разработку программного обеспечения, на самом деле не имел необходимых технических навыков для того, чтобы понять, с какими затруднениями столкнулся программист Джефф. Вдобавок вместо того, чтобы сидеть вместе с Джеффом вечерами и в выходные и убедиться, что работа действительно сделана, он ограничивался телефонными «проверками» Джеффа.

Этот уволенный вице-президент совершил много ошибок. Во-первых, он не ощущал важности окончания работы в срок. Это проистекало из неуважения к деловым обязательствам. «Я сделал все, что мог», – замечательные последние слова, достаточно иррациональные и произнесенные исключительно для самоутверждения. Во-вторых, он был некомпетентен в данной проблеме. Этот вице-президент не знал C, но управлял программистом, который знал этот язык (но, очевидно, не очень хорошо). В-третьих, этот вице-президент не смог определить, в чем настоящая причина проблемы, – в программисте или в программном обеспечении. Он так и не выявил истинных причин надвигающегося бедствия. Итак, итоги вскрытия таковы: его уволили, и рассказанная история была реквиемом по делу, которое он делал.

Что дальше

На протяжении всей этой главы я держал перед вами зеркало (зеркало – метафора самопроверки и самонаблюдения). Глядеться в него было, возможно, неприятно и даже болезненно, но без боли вам не удастся вырасти до руководителя мужчин и женщин, разрабатывающих программное обеспечение в условиях жестких временных рамок. Есть ли секреты в управлении самим собой? Я не знаю ни одного, но точно знаю, что мое определение «сделаю все, что смогу» в контексте руководства программистами имеет другое значение. Бывает, когда фраза «я сделал, что смог» произносится под конец, в случае неудачи. Не думайте об этих словах в таком ключе. Эти слова означают приложение максимума усилий – их можно оценивать каждодневно, успех же иногда приходит только в самом конце. Так что если секрет и есть, то он таков: каждый день проверяйте себя как руководителя и старайтесь стать лучше уже на следующий день. Ваш перечень вопросов к самому себе мог бы выглядеть так:

1. Подвергаю ли я качество своего управления ежедневной оценке?
2. Действительно ли я с каждым днем руковожу все лучше или я постоянно откладываю совершенствование стиля и сути моего управления на потом?
3. Нравится ли мне то, что я делаю?
4. Теряю ли я попусту время при выполнении своих служебных обязанностей?
5. Оцениваю ли я свою производительность тем, сколько сделали мои подчиненные под моим руководством, или у меня есть ощущение, что сам я не сделал ничего?
6. Как мои слабости дали себя знать сегодня (по отношению ко мне самому или другим)?
7. Чему я научился сегодня, чтобы оставаться в курсе, чтобы быть осведомленным, чтобы углубить и расширить свои знания?

Этот перечень содержит семь пунктов; число семь древние считали совершенным. Вы не достигнете совершенства ни сегодня, ни назавтра, ни, возможно, в течение всей вашей жизни, но вы можете сделать максимально эффективными свои усилия, направленные на самосовершенствование, делая все, что нужно, ежедневно.

В следующей главе, в которой мы узнаем, как повести котов за собой, вам придется еще раз взглянуть в зеркало. Пускай самоанализ станет для вас островком безопасности, но не тайником. Выполняя вашу работу, вы должны предвидеть будущее и глядеть по сторонам, но при этом всегда необходимо иметь в виду тот факт, что становление вашего характера руководителя – процесс длиною в жизнь.

Глава 3. Как вести стаю за собой

Навыки руководителя нельзя выработать исключительно силой воли. Для того чтобы они появились, вы должны овладеть некоторыми, возможно, еще неизвестными вам принципами менеджмента. Эта глава как раз и посвящена ряду важных областей менеджмента, которые вам предстоит контролировать, ибо в противном случае они будут контролировать вас. Я намерен познакомить вас с теми аспектами вашей новой работы, которые непосредственно позволяют заставить двигаться ваших программистов в одном направлении – а, как я уже говорил, именно эта задача в процессе выпаса котов является для руководителя основной. Конкретнее, мы поговорим об организационном управлении, пронизывающем всю вашу рабочую деятельность. Я расскажу, что делать с раздражителями, которые имеют обыкновение постоянно отвлекать от работы. Этими навыками вы должны овладеть в обязательном порядке – вне зависимости от того, как они повлияют на ваши отношения с окружающими. Управление проектами – еще одно важное направление деятельности руководителя – рассматривается в этой главе в контексте остальных ваших обязанностей. Мы также поговорим о том, как формировать группы и как работать с персоналом, – теми людьми, которые определяют конкретный результат ваших усилий. Короче говоря, материал, содержащийся в этой главе, совершенно необходим для любого, кто хочет достичь совершенства в деле выпаса котов.

Как справиться с административными функциями

С тех пор как вы стали руководителем, ваше мировоззрение начало кардинально меняться. Будучи программистом, вы привыкли непринужденно общаться с себе подобными. Так вот, имейте в виду, что милые беседы по поводу того, как лучше организовать системное прерывание из-за ошибки, понемногу уходят в прошлое. Теперь вам придется обсуждать тонкости форматирования коммерческих требований, без чего вы просто не сможете составить спецификацию проекта. Конечно, для написания такой документации существует огромное количество разных шаблонов, но проблема-то в том, что для решения этой и множества аналогичных задач, с которыми постоянно сталкивается руководитель, необходимо изменить стиль мышления.

Координация информационных потоков – это тот род деятельности руководителя, который требует отдельного рассмотрения. Как вы, я надеюсь, помните, согласно методике объектно-ориентированного (ОО) анализа и проектирования, для формулирования проектного решения и разрешения проблем, связанных с конструированием программного продукта, вы должны иметь в виду три перспективы. Если хотите освежить память, взгляните на табл. 3.1.

Таблица 3.1. Концепция объектно-ориентированного проектирования

Перспектива	Назначение
Бизнес	Анализ восприятия информации и процессов пользователем
Спецификация	Анализ публичных свойств интерфейса объекта
Реализация	Все внутренние детали объекта, обеспечивающие его функционирование

А теперь попробуйте привязать эту объектно-ориентированную концепцию к вашему повседневному общению с различными подразделениями компании, направленному на решение программных и управленческих задач. Вероятно, у вас получится своего рода программа интеллектуального анализа, похожая на показанную в табл. 3.2.

Таблица 3.2. Административный фильтр

Перспектива	Инструмент	Назначение
Бизнес	Глаза и уши	Сбор информации, необходимой для координирования программных задач
Спецификация	Организация	Систематическое отслеживание ожидаемых от вас результатов (в качестве единицы систематизации можно избрать, например, проект или технологическую область)
Реализация	Углубленный анализ	Дотошное выявление всех подробностей, составляющих ожидаемый результат

Предлагаемый мною административный фильтр в первую очередь призван помочь вам сориентироваться в информационной мгле[31], которая нас ежедневно окружает. Мгла эта состоит из электронных сообщений, телефонных звонков, технических условий, маркетинговых сведений, планов Microsoft на следующий год, вашей вечной головной боли в лице разгильдяя-программиста Джорджа и т. д. Другими словами, пытаясь вести свою стаю, вы ежедневно, если не ежечасно, попадаете в ситуацию информационной перегрузки. Все поступающие све-

[31] Термин «информационная мгла» (data smog) я позаимствовал у Дэвида Шенка (David Shenk).

дения на первый взгляд кажутся важными, однако без фильтра не обойтись. Смысл этого фильтра можно выразить одним словом (кстати, это еще один удачный термин из области объектно-ориентированного программирования):

ФОКУСИРОВКА

Помните ваш первый поход к врачу, когда еще ребенком вы выяснили, что должны носить очки? (Я задаю такой вопрос исходя из того, что вы – главный программист; следовательно, у вас большой опыт борьбы с мелочами жизни, а значит, вам, скорее всего, нужны очки.) Правда, замечательное ощущение, когда, надев линзы, вы понимаете, что теперь оптометрическая таблица вам нипочем? Это и означает умение сфокусироваться – отбросить все раздражающие факторы, которые неизменно сопровождают деятельность руководителя, и выявить первоочередные задачи.

Намотайте себе на ус: если уж у вас есть задача организовать производство программного продукта, остальные сведения, отвлекающие от этой задачи, независимо от того, насколько важными они кажутся, нужно отложить[32].

> Если уж у вас есть задача организовать производство программного продукта, остальные сведения, отвлекающие от этой задачи, независимо от того, насколько важными они кажутся, нужно отложить.

Вот некоторые ситуации-раздражители, часто встречающиеся в повседневной деятельности руководителя.

• Кто-то из числа опытных пользователей вашего продукта придумал *замечательную* новую функцию и считает, что обязан рассказать вам о ней прямо сейчас.

• Сотрудник группы поддержки продукта в очередной раз сообщает вам о том, что количество записей в журнале ошибок с каждой неделей растет в геометрической прогрессии, и потому вы просто обязаны расставить в нем приоритеты.

• Проектировщик продукта испытывает страстное желание узнать, можно ли будет в обозримом будущем реализовать в коде желаемую функцию. По этой причине он пишет очередное письмо с настоятельной просьбой ответить на все предыдущие.

Обратите внимание на слова, выделенные курсивом: «замечательный», «обязан», «просьба». Смахивает на основной принцип подготовки вечерних теленовостей – «чем больше крови, тем лучше». Большинство раздражителей не так важны, как кажутся. Однако если их формулируют с помощью эпитетов типа «замечательный» и «обязан», а также с добавлением просьб, вы, скорее всего, не удержитесь, отвлечетесь от основной задачи и потеряете фокусировку. В этом-то и состоит проблема. Учитесь систематизировать те действия, которые вам рано или поздно придется выполнить, но не забывайте о том, что ваша главная задача – выпустить программный продукт. Иногда человеку, который обратился к вам с просьбой, полезно быстро ответить и, соответственно, дать знать, что вы его просьбу получили. Но при этом ответ должен выглядеть примерно так: «Я обязательно включу вашу проблему в график и позже рассмотрю, но в данный момент у меня очень много обязательств, и пока что я не могу этого сделать».

Какое отношение, спросите вы, все эти разговоры о фокусировке имеют к ведению стаи в нужном направлении? Ну, если вы действительно хотите, чтобы коты за вами шли – а их, как известно, сложно заставить это сделать, – придется продемонстрировать им ваши лидерские качества. Не нужно разглагольствовать о лидерстве – вы обязаны показать программистам решимость выпустить общими усилиями качественный код. Естественно, руководитель, помимо прочего, должен учитывать все тонкости (в частности, те, что перечислены

---

[32] В главе 1 я уже ссылался на труд Стивена Кави (Stephen Covey) под названием «The 7 Habits of Highly Effective People». Если у вас нет этой книги, идите в магазин, купите и прочитайте. Обязательно обратите внимание на то, что Кави думает по поводу организации времени и расстановки приоритетов.

выше), которые помогают укомплектовать код. Фокусироваться – значит расставлять приоритеты среди тех сведений, которые претендуют на значимость наравне со сведениями, действительно значимыми для завершения текущих проектов.

Фокусироваться – значит расставлять приоритеты среди тех сведений, которые претендуют на значимость наравне со сведениями, действительно значимыми для завершения текущих проектов.

Предположим, например, такую ситуацию: в ходе написания кода вы замечаете в одном из объектов, который предполагаете расширить (или добавить интерфейс), некую аномалию. Искушение приняться за исправление такого объекта – по большому счету, попавшегося вам на глаза случайно – велико. Что сделает хороший программист? Он примет аномалию во внимание, но продолжит заниматься текущей процедурой. Вспомните, как просто потерять концентрацию при кодировании, отвечая на телефонный звонок. На то, чтобы в подобных ситуациях вернуться в колею (и восстановить в сознании логику кода), уходит битый час. В организационном управлении действует тот же принцип: либо вы решаете первоочередные задачи, либо теряете последние шансы укладываться в график. Ко времени, выделяемому на решение административных задач, нужно относиться так же трепетно, как и ко времени, выделяемому на кодирование.

Ко времени, выделяемому на решение административных задач, нужно относиться так же трепетно, как и ко времени, выделяемому на кодирование.

Как не отвлекаться на раздражители

Не позволяйте своему почтовому ящику влиять на ваш распорядок дня. С одной стороны, электронная почта – самое полезное в мире изобретение после хлеба в нарезке, но, с другой – есть в ней что-то от нечистого. Как средству оперативного обмена информацией ей нет равных; проблема в том, что в условиях географической рассредоточенности она становится основным носителем информационного потока. Позволив почте влиять на повседневные приоритеты, вы рискуете потерять концентрацию. Разрастание рамок проекта часто начинается с почтового сообщения, отправитель которого пытается уточнить коммерческие требования. Если дискуссия разрастается до трех сообщений, беритесь за телефонную трубку. Если звонок не решает проблему, попробуйте встретиться с собеседником лично. Если и это не принесет желаемого результата – бросайте все попытки его достичь. Главное – чтобы электронная переписка не мешала решать задачи, включенные в график. Если в вашей компании имеется какая-то программа руководства проектами, опирайтесь на эту программу, собирая информацию по конкретным задачам; не загромождайте ее почтовым мусором, поскольку, если только не включать переписку в проектную документацию, она все равно, скорее всего, затеряется.

Еще один дьявольский инструмент уточнения – служба мгновенных сообщений. Пользоваться ею нужно с умом. Не тратьте на нее свое время впустую и не позволяйте ее значку на рабочем столе постоянно отвлекать вас от дела. Не пытайтесь с ее помощью проверять, находятся сотрудники на рабочих местах или нет, – никто не любит, когда за ним открыто наблюдают. В деле слежки нужно проявлять изобретательность – об этом, кстати, я говорил в главе 2 в подразделе «Бессмысленно ожидать чего-либо при отсутствии контроля» раздела «Естественный отбор и время».

Привыкая к решению организационных проблем, вы обнаружите, что раздражители влияют не только на вас – программисты подвержены им не меньше. Одна из главных ваших обязанностей заключается в том, чтобы приучить сотрудников концентрироваться на работе.

Одна из главных ваших обязанностей заключается в том, чтобы приучить сотрудников концентрироваться на работе.

Как справиться с этой задачей? Лучше всего еженедельно назначать всем программистам перечни задач, аннотируя их приоритетами и сроками завершения. Поскольку цикл разработки всегда отличается непостоянством, перечни эти могут меняться каждую неделю, а иногда – даже каждый день. Ответственность за составление и корректировку этих списков ложится, опять же, на вас[33]. Вы себе не представляете, сколько менеджеров считают программистов чуть ли не телепатами и пребывают в уверенности, что график работы они составляют с помощью интуиции. Таких убеждений придерживаются даже некоторые директора. Подобный подход регулярно приводит к тому, что вместо работы во имя пользователей программисты трудятся для программистов, а руководители обнаруживают свою бесполезность в контексте упрочения позиций компании.

Если вы унаследовали персонал от предыдущего руководителя, попросите каждого сотрудника в письменном виде сформулировать его текущие задачи. Это очень эффективный прием – вы не только узнаете, что программисты думают о своих обязанностях, но и составите представление о том, как руководство осуществлялось ранее. Просмотрев перечни задач, составленные самими сотрудниками, вы сможете оптимизировать их функции. Однажды при-

[33] Ряд предложений касательно программных продуктов, помогающих составлять для программистов перечни задач, содержится в главе 4. Но не торопитесь переходить к этому материалу – иначе пропустите все мои подготовительные пассажи о том, в какую путаницу вы рискуете попасть, если не призовете на помощь инструменты электронной организации рабочего процесса.

няв решение о руководстве методом перечней задач, вы не сможете от него отказаться. Все будут ждать новых заданий, и чем последовательнее вы себя проявите в деле их назначения, тем очевиднее станут ваши лидерские навыки. И еще один момент. Перечень задач – это лишь первое звено в процессе ведения стаи в нужном направлении. По меньшей мере раз в неделю, а то и каждый день вам придется доносить до сотрудников дополнительные сведения, инструктировать их и помогать двигаться к намеченной цели, соблюдая при этом установленные временные ограничения.

Скорее всего, вам не удастся сильно продвинуться в вопросах физического размещения программистов, но все-таки при любой возможности требуйте, чтобы каждый сотрудник вместо отгороженной кабины имел отдельное помещение. Если руководство вашей компании больше думает не о продуктивности, а об арендной плате, у вас мало шансов – и, тем не менее, гните свою линию. Один из наиболее заметных раздражителей в деятельности программистов создают инженеры-технологи компаний с их квадратно-гнездовым мышлением. Фильм «Office Space», в котором, так сказать, «в естественном виде» изображены программисты в своих кубышках, должен стать обязательным для просмотра вредителями, планирующими офисное пространство. Факторы, влияющие на «отупение» сотрудников, на примере 32 346 компаний изучили Том Димарко (Tom DeMarco) и Тимоти Листер (Timothy Lister). На составленной ими диаграмме видна четкая обратная зависимость между степенью отупения сотрудников и объемом выделяемого каждому из них офисного пространства[34]. О чем это говорит? О том, что шум, отвлекающие факторы и все прочие побочные эффекты политики снижения затрат серьезно снижают продуктивность работы.

Так пусть ваши программисты работают дома – это один из лучших способов исключить раздражающие факторы и поднять продуктивность. Но и здесь не все так просто! Некоторым сотрудникам такой режим работы подходит лучше всего; другим полезно появляться на работе хотя бы раз в неделю; в любом случае, эффективной деятельности в режиме «на дому» достигают только самые дисциплинированные. Выбрав этот путь, вы будете вынуждены уделять довольно много времени проверкам. Опытным и надежным сотрудникам можно доверить работу на дому; что касается остальных – дайте им возможность попробовать, но обязательно проверяйте новые показатели продуктивности. Если отделения компании, в которой вы работаете, рассредоточены по всему земному шару, и у вас фактически нет выбора, будьте готовы к тому, что с прижатой к уху телефонной трубкой придется проводить по 10 часов в неделю, а от электронных сообщений не будет отбою. Схема работы на дому очень перспективна, но чтобы заставить ее приносить плоды, вы должны учесть все тонкости.

[34] DeMarco and Lister, op. cit., p. 56.

Когда проект разрастается

Я имею в виду классическое понятие разрастания проекта. Если аналитик бизнес-требований утверждает, что занимается уточнением рамок, знайте: он их раздвигает. Не важно, как этот процесс называть. Факт тот, что специфицировать и сконструировать программу, избежав разрастания требований, не удавалось еще никому. Объясняется это натурой человека – невозможно с первой попытки сформулировать все функции, которые предполагаемой программе предстоит выполнять.

Рассмотрим типичный процесс производства программного продукта. Вне зависимости от того, что вы предпочитаете – стремительный напор или постепенные итерации, – процесс этот всегда состоит из нескольких фиксированных этапов.

1. *Замысел*. У кого-то появляется блестящая идея.

2. *Специфицирование*. Куча людей пытаются описать эту идею.

3. *Проектирование*. Высокоинтеллектуальные товарищи решают, как сконструировать предполагаемый программный продукт.

4. *Конструирование*. Бессонными ночами и нескончаемыми днями программисты программируют.

5. *Тестирование*. Обнаруживается, что конечная реализация блестящей идеи не так хороша, как хотелось бы, или, еще хуже, что идея-то отнюдь не блестящая.

6. Все начинается заново со второго этапа, пока вы не почувствуете, что все нормально, всего хватает, или, наоборот, не придете к заключению, что идея, высказанная на первом этапе, ужасна и вам срочно требуется новый блестящий замысел (в последнем случае, опять же, все начинается со второго этапа).

А теперь подумайте: таким ли уж неожиданным станет разрастание рамок в контексте отраженного в этом списке реального сценария? Мне так не кажется, и вам тоже не должно так казаться. И тем не менее, если речь об этом зайдет, воя и зубовного скрежета программистов не избежать. Как заставить их поверить, что вы во всеоружии и ваша позиция правильна? Лучше всего поставить их перед фактом, что разрастание неизбежно, и научить их справляться с этой проблемой. Вы руководитель, и это – ваша обязанность. Не поддавайтесь соблазну разнести «крайних» в других отделах – этим вы не приблизите завершение работы и не исправите ситуацию.

В своей немного устаревшей, но, тем не менее, сохраняющей значимость книге под названием «Managing the Software Process» Уотс Хамфри (Watts Humphrey) сформулировал принцип, актуальный по сей день:

«Когда программисты берутся за оценку объема кода реализации какой-либо функции, результаты неизменно оказываются заниженными. Этому есть множество возможных объяснений. В этом контексте следует понимать, что их оптимизм есть относительно прогнозируемая функция состояния проекта»[35].

На самом деле, разрастание рамок объясняется очень просто – сказать, сколько времени и сил уйдет на создание очередной сногсшибательной программы, вплоть до ее первого выпуска и критической оценки, не может никто. Многие программисты соглашаются с двумя соображениями относительно проводимых ими первоначальных оценок, согласующихся с принципом Хэмфри; я сформулирую их в виде аксиом:

- любой процесс продлится дольше, чем вы надеетесь;
- всегда появляется что-то, о чем вы не подумали.

[35] Watts S. Humphrey, Managing the Software Process (New York: Addison-Wesley, 1989), p. 93.

Вооружившись этими аксиомами и введенным Хэмфри понятием «состояния проекта», старайтесь контролировать разрастание и обязательно убедите ваших подчиненных в том, что вы человек проницательный, поскольку предсказывали такую возможность и учли ее в процессе тщательного планирования. «Тщательное планирование»! Звучит замечательно, но что же это означает в контексте выпаса котов? А вот что. Вернитесь к главам 1 и 2, еще раз пробегитесь по изложенным в них принципам и попытайтесь сделать их неотъемлемыми элементами своего характера. Помните, что лидерство происходит от сердца, а не от ума[36].

Конечно, и для мозга найдется работа – в частности, ему предстоит разработать методы контроля над разрастанием рамок проекта. Давайте рассмотрим типичный план проекта. Предположим, что, исходя из заданного набора требований, вы должны разработать проектное решение с расчетом на его последующую реализацию программными средствами. Элементарный, но наивный план иллюстрирует табл. 3.3.

Таблица 3.3. Нереалистичный план проекта

Задача	Время выполнения (произвольные интервалы)
Анализ требований	A
Создание проектного решения	B
Реализация проектного решения	C
Тестирование программного обеспечения	D
Исправление ошибок	E
Развертывание программного обеспечения	F

Руководствуясь таким планом, вы рискуете нарваться на кучу неприятностей. Будучи глубоко убеждены, что конечная дата сдачи проекта будет равняться сумме временных интервалов A + B + C + D + E + F, вы немало удивитесь, обнаружив, что это совершенно не так.

Рассмотрим более реалистичный план, показанный в табл. 3.4.

Таблица 3.4. Реалистичный план проекта

Задача	Время выполнения (произвольные интервалы)
Анализ требований	A
Обсуждение результатов анализа с сотрудниками отдела	B
Создание проектного решения	C
Макетирование проектного решения	D
Оценка макетов	E
Пересмотр проектного решения	F
Реализация высокоуровневых объектов проектного решения	G
Тестирование высокоуровневой интеграции	H
Оценка системы на предмет соответствия требованиям	I
Создание компонентов системы	J
Интеграция и тестирование компонентов	K
Повторная оценка системы на предмет соответствия требованиям	L

[36] Вспомните, что говорил Йода: «Сила Джедая есть результат его собственных усилий». У нас то же самое.

Задача	Время выполнения (произвольные интервалы)
Тестирование комплектной системы	M
Исправление неисправностей системы в преддверии альфа-тестирования	N
Начало альфа-тестирования	O
Исправление ошибок, выявленных на этапе альфа-тестирования	P
Начало бета-тестирования	Q
Разработка стратегии развертывания	R
Исправление ошибок, выявленных на этапе бета-тестирования	S
Тестирование стратегии развертывания	T
Тестирование конечного продукта	U
Развертывание программного обеспечения	V

Проводить арифметические операции я, пожалуй, не буду, поскольку в таблице мне еле хватило букв латинского алфавита. В общем, вы все поняли. Помните также, что приведенный план не учитывает тех индивидуальных этапов, которые обусловливаются особенностями проекта, равно как и зависимости между различными этапами цикла разработки. В то же время он успешно иллюстрирует причины разрастания рамок проектов, позволившие мне сформулировать две вышеупомянутые аксиомы: это, во-первых, недостаточный анализ подробностей, и, во-вторых, излишне оптимистические оценки длительности конструирования программных продуктов. Чем больше опыта вы наработаете в деле выявления подобных деталей, тем точнее вы сможете оценить время работы над проектом и тем больше вы будете убеждаться в том, что разрастание рамок проекта происходит в результате недоработок специалистов, отвечающих за планирование.

> По большей части, разрастание рамок проекта происходит по причине недоработок специалистов, отвечающих за планирование.

Каков механизм совершенствования навыков временной оценки? Он очень прост – поначалу вам предстоит неоднократно нарушать утвержденные графики сдачи проектов, но, в конце концов, вы научитесь им соответствовать. Практика эта довольно нервозная и даже угрожающая вашему карьерному росту. Возможно, это не лучший способ самосовершенствования, но что можно утверждать со всей определенностью, так это то, что в области управления проектом опыт играет огромную роль. А чтобы ускорить обучение, почитывайте анекдоты, относящиеся к руководству проектами. В своей леденящей душу коллекции очерков о неудавшихся программных проектах Роберт Гласс (Robert Glass) составил список наиболее распространенных «программных катастроф», который я привожу в порядке снижения значимости[37].

1. Неадекватное специфицирование задач проекта (51 %).
2. Неудовлетворительные планирование и оценка (48 %).
3. Применение новой для данной компании технологии (45 %).
4. Негодная/отсутствующая методология руководства проектом (42 %).
5. Нехватка ведущих специалистов группы (42 %).
6. Срыв договоренностей производителями аппаратного/программного обеспечения (42 %).

Процентные показатели, приведенные для каждой из вышеупомянутых причин несостоятельности, выведены Глассом в ходе специального исследования с целью выявить основную причину выхода процесса разработки программных продуктов из-под контроля. Я очень реко-

[37] Robert L. Glass, Software Runaways (Upper Saddle River, NJ: Prentice Hall, 1998), p. 20.

мендую ознакомиться с этой и другими подобными работами[38]. В результате вы поднимете процесс обучения на более осмысленный уровень, получите определенное представление о реалистичном планировании проекта, овладеете методикой контроля над разрастанием рамок проекта.

[38] См. также Robert L. Glass, ComputingFailure.com (Upper Saddle River, NJ: Prentice Hall, 2001).

Как объединить усилия тех, кто гуляет сам по себе

Речь не о кошках, а о программистах. О том, что они блуждают в потемках, можно судить по характеру их кода, – он начинает походить на огромный памятник их гениальности. Практика регулярного критического обзора кода помогает сносить подобные монументы еще до того, как самовлюбленные хозяева их окончательно возведут. Более подробно мы побеседуем об этом явлении в главе 6, полностью посвященной техническому руководству. Пока что запомните один замечательный принцип: творчество бесценно, а вот практичный и удобный в сопровождении код можно не только оценить, но и продать. В ваши обязанности как руководителя входит координация деятельности программистов, направленная на достижение максимальной функциональности за счет минимального объема кода. Усложнение – это то, с чем вам предстоит вести постоянную борьбу. К понятности кода придется идти мелкими шажками – благо, скорее всего, вам предстоит бороться со смертными грехами программистов, такими как[39]

[39] Это лишь немногие грехи. Существует множество альтернативных перечней. Добротный пример см. в издании William H. Brown et al, AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis (New York: John Wiley & Sons, 1998).

Конец ознакомительного фрагмента.