Agile
менеджмент

Лидерство
и управление
командами

Юрген Аппело

Эта книга предназначена
для менеджеров, которые хотят перейти
на гибкие методы управления в своих
компаниях, и на разработчиков, которые
уже используют эти методы при
создании ПО, но хотят больше узнать
о менеджменте в целом.

альпина паблишер | бизнес

Юрген Аппело
Agile-менеджмент. Лидерство и управление командами

Текст предоставлен правообладателем
http://www.litres.ru/pages/biblio_book/?art=35478823
Agile-менеджмент: Лидерство и управление командами / Юрген Аппело: Альпина Паблишер; Москва; 2018
ISBN 978-5-9614-0937-6

Аннотация

Во многих организациях на пути внедрения Agile оказывается традиционный менеджмент. Командам тяжело применять гибкие методологии, если их лидеров заклинило на устаревших управленческих подходах.

Цель этой книги – дать понять, как работают Agile-команды. В ней нет кейсов, простых решений и банальных советов. Чего в ней в избытке, так это интересных идей, результатов экспериментов и поводов для размышления. В ней есть то, что действительно необходимо современным менеджерам: понимание общих подходов, с помощью которых вы сможете создать собственные рецепты, соответствующие именно вашим потребностям.

Содержание

Юрген Аппело
Agile-менеджмент. Лидерство и управление командами

ЮРГЕН АППЕЛО

Agile-МЕНЕДЖМЕНТ

Лидерство и управление командами

Перевод с английского

альпина
ПАБЛИШЕР

Переводчик *А. Олейник*
Научный редактор *Анна Обухова*
Редактор *А. Черникова*
Главный редактор *С. Турко*
Руководитель проекта *А. Василенко*
Корректоры *Е. Аксёнова, О. Улантикова*
Компьютерная верстка *А. Абрамов*
Дизайн обложки *Ю. Буга*

Аппело Ю.
Agile-менеджмент: Лидерство и управление командами / Юрген Аппело; Пер. с англ. – М.: Альпина Паблишер, 2018.

ISBN 978-5-9614-0937-6

* * *

Посвящается Раулю.

С благодарностью за десять лет в одной команде

Предисловие

Роберта Мартина

Я ненавижу книги по менеджменту. Просто ненавижу. Люди все время дают их мне со словами: «Вы должны прочесть эту книгу, она изменила мою жизнь!» В таких книгах обычно порядка 150 страниц, они набраны крупным шрифтом с двойным межстрочным интервалом, и в них много иллюстраций. Они имеют названия вроде: «Как управлять не управляя», «Менеджмент с открытыми дверями», «Сначала нарушьте все правила», «Откройте свои сильные стороны», «Сила позитивных наказаний» или даже «Tnemeganam!». Эти книги только занимают место на моих полках. Иногда я читаю их в туалете.

Все они рассказывают одну и ту же историю. Автор – всегда какой-то парень, которому приходится управлять компанией, которая вот-вот обанкротится. Когда эта компания оказывается совсем в заднице (помните, что такие книги я читаю в туалете?), его вдруг посещает чертовски важное озарение, которое до этого никогда никого не посещало. Когда он описывает свою идею коллегам, те думают, что он рехнулся. Невзирая на это, он внедряет свою идею и в резуль-

тате зарабатывает 1 000 000 000 000 (один триллион) долларов – миллиардом в наше время никого не удивишь. И теперь по доброте душевной готов за небольшую плату поделиться своей идеей с вами, чтобы вы тоже смогли заработать свой триллион.

Такие книги, как правило, страдают повторами, наивны, бессодержательны и написаны на уровне третьего класса для старательных двоечников, которые считают, что *одной простой идеи* достаточно, чтобы решить все их проблемы.

Эти несчастные простаки надеются, что если они прочитают последний бестселлер под названием «Стратегия голубых штанов» и заставят всех в офисе носить голубые штаны по четвергам, то их проблемы в области менеджмента будут решены.

Как сказано выше, я ненавижу книги по менеджменту. Так что же заставило меня написать предисловие именно к этой книге? Я делаю это потому, что тут встречается слово «эукариоты»! Что оно означает? Не так важно. Главное, что в этой книге используются научные термины! В ней говорится о *гонке Черной Королевы* и есть изображения *тессерактов*. Тут можно прочитать про *путь пьяницы*. Короче говоря, это *интеллектуальная* книга!

Просто взгляните на оглавление. Вы увидите, что там заявлены такие темы, как *теория сложности, теория игр, кибернетика, самоорганизация* и *принцип темноты*, а диапазон вопросов, о которых пишет автор, чрезвычайно широк –

от определения оптимального размера команды и проблем мотивации до управления масштабированием в сравнении с уплощением.

Когда вы будете читать эту книгу, то убедитесь, что автор хорошо разбирается в своем предмете. Ее содержание ничего общего не имеет с пересказом старых баек о том, как какого-нибудь бывшего футболиста назначили руководить тонущей компанией и он сумел вывести ее в лидеры рынка. Скорее эта книга представляет собой серьезную компиляцию идей управления, методов и дисциплин, накапливавшихся в течение ста с лишним лет. Автор взял эти идеи и связал их с гибкими методологиями разработки программного обеспечения, создав *мемплекс* – взаимосвязанную систему идей, которая нужна каждому, кто сколько-нибудь серьезно изучает менеджмент. Эта книга не для тех, кого интересуют быстрые решения. Она для серьезных читателей, которые глубоко интересуются менеджментом и хотят овладеть его тонкостями.

Эда Юрдона

Давным-давно, в далекой-далекой галактике мы с коллегами с гордостью провозгласили себя молодыми революционерами компьютерной индустрии, положившими начало новому поколению методов и технических приемов программирования, дизайна и анализа программных продуктов. То-

гда нам казалось, что эти методы вполне гармонично сочетаются с директивными управленческими подходами сверху вниз, господствовавшими в то время. Нам не хватило мозгов, чтобы придумать для своих идей название вроде «Программное обеспечение 2.0», как это сделали впоследствии приверженцы «Web 2.0» и «Предприятия 2.0»… Но как бы то ни было, книга Юргена Аппело убедила меня в том, что идеи, выдвинутые моим поколением, оказались на свалке истории.

Проблема здесь не в методах разработки ПО, и книга Юргена на самом деле не о разработке программных продуктов – хотя гибкие методологии за последние десять лет становятся все более популярными и начисто отвергают идею о том, что функциональность и архитектура сложных систем могут быть разработаны строго линейными методами, базирующимися на иерархическом детерминистском подходе сверху вниз. В сложном мире, где конечные пользователи не совсем уверены, чего они хотят от программного продукта, а среда, в которой они работают, изменяется в процессе разработки ПО, нам необходим упорядоченный (смею ли я сказать «структурированный»?) подход к разработке программных продуктов – и все равно многие детали любого проекта остаются неизвестными и непредсказуемыми, если только эмерджентный подход не позволяет выявить их в нужное время.

Если это верно относительно технических функций, таких как анализ, проектирование и внедрение систем, – а я

твердо верю, что это так, – то также это верно и относительно управленческого подхода в целом, который организует, мотивирует, отслеживает, ограничивает и (надеюсь) вознаграждает людей, делающих эти технические задачи.

Таким образом, иерархический стиль управления сверху вниз, который соответствовал нашему иерархическому «структурированному» подходу к анализу и проектированию ПО в 1970-е годы, в настоящее время называют «Менеджмент 1.0». Юрген также сообщает нам, что уже пройдена фаза, известная как «Менеджмент 2.0», которая в значительной степени была представлена новомодными изобретениями типа «Реинжиниринга бизнес-процессов», шести сигм и прочими дополнениями к предшествовавшему им Менеджменту 1.0.

Менеджмент 3.0, который стал предметом этой книги, основан на теории сложности. Это то, чем на протяжении последних нескольких десятилетий занимались математики и биологи. Теперь эта теория становится центральной частью экономики и социологии – а в более общем плане и частью науки об управлении людьми и их взаимоотношениями в организации. Вам действительно стоит прочитать содержащийся в этой книге обзор данной теории и связанных с ней концепций, включая причинно-следственные связи, детерминизм и редукционизм – темы, хорошо знакомые каждому инженеру, математику и специалисту в области компьютеров (все они знакомятся с этими идеями достаточно рано в про-

цессе обучения).

Основываясь на этом фундаменте, вы будете готовы воспринять продвигаемую Юргеном модель современного менеджмента, которая у него представлена в виде шестиглазого монстра, чей взгляд направлен на людей, выравнивание, настройку ограничений, улучшение, развитие компетенций и вопросы структурирования организаций. Вам предстоит продраться через две вводные главы, в которых Юрген дает краткое изложение сути гибких, или Agile-методологий разработки программных продуктов, а также теории сложности; затем он посвящает по две главы каждому из шести компонентов модели Менеджмента 3.0.

Вы не найдете в книге ни одного «традиционного» аспекта управления проектами вроде управления рисками, оценки, планирования или мониторинга процесса разработки с помощью Microsoft Project – он в книге вообще не упоминается. Вы не найдете никаких ссылок на стандартные учебники по управлению рисками, планированию и бюджетированию проектов. Эти традиционные виды деятельности по-прежнему нужны, и, вероятно, имеет смысл пройти курс проектного менеджмента и убедиться, что вы его понимаете, но смысл аргументации Юргена состоит в том, что, даже если вы все будете делать правильно с точки зрения традиционного управления проектами, это совершенно не гарантирует вам успеха. (И даже наоборот, может лишь усугубить проблемы, связанные с поведением сложных систем, что приве-

дет вас к катастрофе еще быстрее!)

Вы можете читать отдельные главы книги Юргена независимо друг от друга, возможно, даже в любой последовательности, но я бы рекомендовал делать это по порядку и усваивать материал постепенно. Книга содержит огромное количество дельных рекомендаций, практичных чек-листов и мудрых советов (как Юргену вообще удалось в своем возрасте обрести всю эту мудрость?) о нюансах лидерства, мотивации, коучинге. А также об общении с отдельными разработчиками, проектными командами и менеджерами, находящимися на более высоких уровнях в организационной иерархии, которые часто так и застревают в устаревших способах управления (эти менеджеры склонны в разговорах называть сотрудников своей компании «ресурсами»). Не исключено, что некоторые из утверждений автора покажутся вам поверхностными из-за своей краткости (вроде констатации в главе 4, что инновации реализуются только снизу вверх и их невозможно внедрить в приказном порядке сверху). Но если вы внимательно прочитаете книгу, то увидите, что она содержит хорошо проработанный материал, а в рассмотрении проблем учтена масса нюансов, как, например, при обсуждении баланса между самоорганизацией и анархией.

Меня позабавило следующее утверждение Юргена почти в самом начале: «Хотелось бы мне, чтобы подобная книга попала мне в руки десять лет назад, когда я занимался своим стартапом. Но в этом случае вполне *могло случиться*, что

я все же заработал бы свои миллионы и, по всей вероятности, вряд ли стал бы заморачиваться написанием этой книги». Меня посетила такая же мысль: было бы крайне полезно, если бы такая книга была доступна (или известна) сорок пять лет назад, когда я впервые начал заниматься разработкой ПО, ну или по крайней мере двумя годами позже, когда меня необдуманно повысили и я стал проектным менеджером. Но в этом случае я тоже мог стать миллионером и вряд ли написал бы это предисловие.

Если серьезно, то единственная реальная проблема, которую я предвижу в связи с этой книгой, эта: менеджеры моего поколения все еще живы, а недавний финансовый кризис обесценил пенсионные программы и заставил этих менеджеров продолжать работать, делая все возможное, чтобы по-прежнему навязывать своим подчиненным жесткий иерархический стиль управления сверху вниз. Еще одна проблема заключается в том, что многие менеджеры поколения, к которому принадлежит Юрген, постепенно продвигаются по иерархической лестнице и начинают занимать высокие позиции – но и среди них немало тех, кто в свое время подвергся промыванию мозгов и долгое время практиковал иерархический подход к менеджменту. Не исключено, что эти менеджеры также будут сопротивляться идеям Менеджмента 3.0.

И тем не менее, если судить по растущей популярности гибких методов разработки ПО, остается лишь немного подождать, чтобы продвигаемые Юргеном Аппело методы об-

щего менеджмента стали столь же популярны. Очевидно, что если вы решите стать «гибким менеджером» и справляться с современными постоянно усложняющимися проектами, то эта книга будет далеко не единственной, которую вам предстоит прочитать на эту тему.

Что еще более важно, вы будете возвращаться к этой книге еще не раз. Я абсолютно уверен, что «Agile-менеджмент» минимум на десятилетие станет библией среди других книг по гибкому менеджменту.

Благодарности

Спасибо. Это, пожалуй, единственное слово, которое никогда не бывает лишним, неуместным или бесполезным. О нем часто забывают. Но только не на этот раз.

Спасибо Майку Кону за то, что он читал мой блог и предложил стать шестым автором в издаваемой им серии книг, а также за оперативные ответы на мои вопросы (иногда в течение часа).

Спасибо авторам других книг, изданных в той же серии, – Лиссе Эдкинс, Лизе Криспин, Джанет Грегори, Клинтону Киту, Роману Пиклеру и Кенни Рубину – за возможность почувствовать себя частью команды и за то, что вы поделились со мной своим опытом – это позволило мне не сделать слишком много ошибок.

Спасибо первым рецензентам моей книги, среди которых Эндрю Вудворд, Анджело Анолин, Кори Фой, Дэвид Харви, Дэвид Моран, Диана Ларсен, Эстер Дерби, Флориан Хоорнаар, Джеффри Лоуни, Израэл Гат, Дж.-Б. Райнсбергер, Якопо Ромеи, Джаред Ричардсон, Йенс Шаудер, Джим Хайсмит, Джоанна Ротман, Джон Бауэр, Келли Уотерс, Лиза Криспин, Луис Дитворст, Марчин Флориан, Маркус Андрезак, Мендельт Сибенга, Майк Кон, Майк Коттмайер, Нико ван Хемерт, Олав Маассен, Пол Клипп, Пол Шталенхоф, Павел Бродзински, Филипп Гадир, Раду Давидеску, Рамкумар КБ,

Роберт ван Куутен, Рассел Хили, Рууд Кокс, Скотт Дункан, Стивен Хилл, Васко Дуарте, Ив Ануй и Закари Спенсер. Ваши ценные (а иногда и болезненные для меня) комментарии помогли сделать эту книгу и ее сайт-компаньон намного лучше. Иногда я даже был согласен с вами.

Спасибо, Крис Гузиковски, Райна Чробак, Шери Кейн, Энди Бистер и все остальные талантливые сотрудники издательства Addison-Wesley, за ваше терпение в работе с начинающим автором и ваши объяснения, как устроен издательский процесс (хотя вам, наверное, приходилось это объяснять в тысячный раз).

Спасибо Стефану Мейджеру, Леннерту Уверкерку, Раджу Менону и другим друзьям, коллегам и знакомым за помощь в ходе написания этой книги. Много маленьких услуг внесли в итоге огромный вклад.

Миссис Стапперс, спасибо за то, что вы научили меня английскому языку. К счастью, существуют онлайн-словари, которые часто выручали меня, когда я не успевал выучить заданные на дом слова.

Спасибо, друзья мои: Амнон, Флорис, Эрик, Фемке, Надира, Девика, Руди, Нильс, Ханнеке, Труди, Йерун и Арно. Редко можно найти людей, готовых искренне поддержать энтузиазм другого человека.

Спасибо моим бывшим коллегам по ISM eCompany. В течение семи лет у меня была возможность учиться, как (не) надо управлять командами разработчиков ПО. Приношу из-

винения, если написанный мною код был из рук вон плох, а также за злоупотребление электронной почтой.

Спасибо, Алистер Коберн, Артем Марченко, Брайан Марик, Кристофер Авери, Кори Хейнс, Деннис Стивенс Эд Юрдон, Элизабет Хендриксон, Джордж Динуидди, Джозеф Пелрайн, Карл Скотленд, Майк Виздос, Филипп Крутен, Рон Джеффрис и многие-многие другие блогеры и авторы, с которыми я имел удовольствие встречаться лично. Все вы вдохновляли меня, и общение с вами было чрезвычайно полезно для этого странного нового «парня на районе».

Спасибо Эду Юрдону и Бобу Мартину за поддержку автора-новичка и написанные вами предисловия. Когда-нибудь я отплачу услугой за услугу. (Дайте знать, если вам вдруг понадобится нарисовать на кого-нибудь карикатуру.)

Спасибо читателям моего блога и тем, кто следит за мной в Twitter. Ваша постоянная поддержка, вопросы и ответы помогли мне пройти этот путь до конца.

Спасибо, Рауль, за предоставленные мне пространство и время, позволившие написать эту книгу. Система может самоорганизоваться только в рамках определенных границ. Я уверен, что мой проект смог вырасти и расцвести только благодаря тем мягким ограничениям, которыми я тебе обязан.

И спасибо вам, уважаемый читатель, что вы открыли эту книгу. Если она вам понравится, пожалуйста, дайте мне знать. А если нет, то сообщать мне об этом не надо.

Об авторе

Юрген Аппело – автор, спикер, тренер, разработчик, предприниматель, менеджер, читатель, блогер, лидер, мечтатель и свободный философ. К тому же он голландец, что объясняет его многочисленные странности.

После изучения программирования в Делфтском техническом университете и получения в 1994 году степени магистра Юрген занимался созданием стартапов и руководил

несколькими голландскими компаниями в роли лидера команд, менеджера и топ-менеджера.

Его последнее место работы – директор по информационным технологиям в ISM eCompany, одном из крупнейших поставщиков решений для электронной коммерции в Нидерландах. В качестве менеджера Юрген возглавлял группы разработчиков программного обеспечения, проектных менеджеров, менеджеров по качеству и сервису, некоторых из которых он нанял случайно.

Среди его основных интересов – разработка программного обеспечения и теория сложности с точки зрения менеджера. В качестве автора он публиковал аналитические работы и статьи во многих журналах, а также ведет блог на сайте http://noop.nl. Его часто приглашают выступать на семинарах и конференциях.

И последнее (не по важности) его достижение: Юрген проводит обучающие семинары на основе модели Менеджмента 3.0, где рассматриваются темы развития инициативы, расширения полномочий и возможностей команд, оптимизации ограничений, развития компетенций, развития организационных структур и улучшения всего.

Тем не менее иногда он откладывает в сторону написание статей, выступления и проведение тренингов и сам занимается программированием; дома он проводит время, сортируя свою коллекцию книг по научной фантастике и фэнтези: он складывает их в шкаф высотой четыре метра, который

сам и сконструировал.

Вместе со своим партнером Раулем Юрген живет в Роттердаме (Нидерланды) – и иногда в Брюсселе (Бельгия). У него двое детей, а также есть воображаемый хомяк, которого зовут Джордж.

Предисловие автора

Это книга о **гибком, или Agile-менеджменте** – управленческом аналоге гибких методологий разработки ПО. Я считаю, что гибкий менеджмент недостаточно представлен в мире, который требует гибких подходов. Существуют десятки книг по Agile-методологиям для разработчиков, тестировщиков, коучей и проектных менеджеров, но практически нет книг по этой тематике для Agile-менеджеров и лидеров команд. Но, если организации хотят внедрить Agile-практики, абсолютно необходимо, чтобы лидеры команд и другие руководители знали лучший подход для управления и лидерства в их командах.

Исследования показывают, что при переходе к гибким методам основным препятствием оказывается традиционный менеджмент [VersionOne 2009]. Командам разработчиков ПО тяжело внедрять такие процессы, как Scrum, XP или канбан, если их «лидеров» заклинило на устаревших управленческих подходах. Менеджерам необходимо понять, в чем заключается их новая роль в XXI веке и как добиваться от команд разработчиков максимальных результатов. Данная книга предназначена для менеджеров, которые хотят перейти на гибкие методы управления в своих компаниях, и на разработчиков, которые уже используют эти методы при создании ПО, но хотят больше узнать о менеджменте в целом.

Эта книга по менеджменту уникальна, поскольку целиком основана на научном подходе и теории сложности. В отличие от других книг по общему менеджменту, она не призывает вас открыть свое сердце, взяться за руки и повторять мантры. Многие менеджеры, особенно в высокотехнологичных компаниях, предпочитают пользоваться левым полушарием мозга, полагаясь на рациональное, аналитическое начало. Поэтому я написал книгу, апеллирующую к таким людям. Но и тем, кто предпочитает пользоваться правым полушарием, нечего опасаться. Научные идеи представлены достаточно неформально, с подробными объяснениями и обилием метафор и иллюстраций. Здесь даже можно найти как минимум пару действительно смешных шуток.

Одной из моих важнейших целей при написании этой книги было придерживаться *описательного* подхода, а ни в коем случае не *нормативного*. Цель – дать вам понять, как работают организация и Agile-команды для того, чтобы вы могли решить собственные проблемы. Мир слишком сложен, чтобы можно было отделаться списком практик, которым необходимо следовать. Что действительно необходимо менеджерам в XXI веке, так это понимание общих подходов, используя которые, они смогут создать свои собственные рецепты, соответствующие их конкретным потребностям [Mintzberg 2004: 252].

История этой книги

Мне потребовалось десять лет, чтобы написать эту книгу. В свое время я заинтересовался гибкими методологиями разработки ПО и теорией сложности (не помню, в какой последовательности), и в течение первых пяти лет авторы, пишущие об этих двух предметах, едва поспевали за моим интересом. При чтении разных книг у меня постепенно начала складываться общая картина. Я понял, что гибкие методы создания ПО – это практическое приложение теории сложности и команды разработчиков ПО и соответствующие проекты выступают в качестве примера таких систем. Также стало ясно, что практически никто не видит эту связь между теорией и практикой (заметными исключениями стали Джим Хайсмит и Кен Швабер). В результате примерно в 2005 году я попытался написать собственную книгу на эту тему. Но в тот момент ничего не получилось. У меня был в руках текст, но отсутствовали читатели. Были новые идеи, но не было обратной связи. Обилие теорий и минимум опыта. Я был преисполнен энтузиазма, но мне не хватило терпения.

Параллельно все эти десять лет я занимался управлением проектами по разработке ПО и приобрел обширный опыт, узнал о множестве способов неправильного управления проектами. Будучи руководителем и внедряя гибкие методологии разработки, я размышлял о роли менеджмента в этом

процессе. Я был уверен, что менеджерам и лидерам команд должна отводиться важная роль. Но в книгах ничего не говорилось о том, в чем конкретно она должна состоять.

В январе 2008 года я запустил свой блог на http://noop.nl с целью получить обратную связь от читателей относительно моих идей в области разработки ПО, менеджмента и сложных систем, а также понять, интересна ли эта тематика вообще кому-нибудь. Через полтора года у меня было 4000 подписчиков. Я участвовал в интереснейших дискуссиях с экспертами со всего мира и удачно выступил на нескольких конференциях в Европе и США. Было похоже, что я нашел свою нишу.

В августе 2009 года, уже после глобального финансового кризиса, я подумал, что пришло время сделать вторую попытку написать книгу. На этот раз все было очень легко. У меня был архив моего блога, полезная обратная связь от читателей, десять лет менеджерского опыта (в основном отрицательного), полно времени (дела шли неважно) и достаточное количество подписчиков в блоге, чтобы мотивировать нескольких издателей предложить мне контракт на книгу. После подписания первого в моей жизни договора в качестве автора все, что мне оставалось сделать, – это удвоить свои усилия в части исследований, утроить интеллектуальные усилия и в четыре раза увеличить свою продуктивность как автора. (Все это звучит гораздо проще, чем было на самом деле.)

Вы, конечно, обратили внимание, что я не являюсь ни консультантом по Agile-методологиям, ни ученым, специализирующимся в области сложных систем. В этом моя сила – и моя слабость. Сила в том, что я редко страдаю туннельным зрением. Мое мышление не испорчено пристрастием к каким-либо конкретным научным подходам, методам или предпочитаемым по умолчанию решениям. Еще со школы мне удавалось видеть общие закономерности и аналогии между различными предметными областями, и учителя в свое время советовали мне выбрать карьеру, которая имела бы отношение к анализу проблем. Моя слабость в том, что часто я воспринимаю проблему со слишком большой высоты. Мне не хватает детальных знаний, которые есть у ученых, и глубокого опыта, который имеется у консультантов, изучивших изнутри множество компаний. Зато мне повезло, что я сумел развить у себя способность писать простые, неожиданные, конкретные, убедительные и эмоциональные тексты. В конце концов, неидеальное, но хорошо написанное послание лучше, чем идеальное послание, которое никто не хочет читать.

Пока я писал эту книгу, я использовал свой блог для получения откликов от подписчиков, так что они не давали мне сбиться с пути, помогали точнее мыслить и отсеивать не слишком полезные идеи. В результате получилась именно та книга, которую я в течение десяти лет мечтал написать. Но в определенном смысле это и та книга, которую хотели уви-

деть мои читатели.

Структура книги

В этой книге вы не найдете конкретных кейсов или пространных списков «стандартных» подходов. Вместо этого здесь приводятся результаты исследований, метафоры, идеи и поводы для размышлений. Все это не делает книгу менее полезной. Напротив, существует точка зрения, что самые значительные достижения осуществляются путем копирования идей из одной области и адаптации их к другой области. Стратегии выживания в биологических экосистемах могут научить не хуже, чем кейсы из практики других компаний. Идеи редко идеально соответствуют вашей ситуации. Именно вы должны решать, могут ли эти идеи применяться конкретно в вашем случае и если могут, то как.

Пользоваться книгой просто. Начните с начала. Потом читайте и переворачивайте страницы. Закончив читать страницу, переверните ее и начните следующую. В один прекрасный момент вы уткнетесь в пустую страницу. Это будет конец книги.

Глава 1 – *введение*. В ней описано, как линейное мышление порой приводит нас к неправильным выводам. Здесь же впервые представлена центральная модель книги: *Менеджмент 3.0* – шесть углов зрения, с которых в ней рассматриваются команды и организации.

В главах 2 и 3 содержится *обзор гибких методологий раз-*

работки программного обеспечения и *теории сложности*. Они представляют собой две основы гибкого менеджмента и тех шести компонентов модели, о которых идет речь в последующих главах.

В главах 4 и 5 описывается первый компонент модели Менеджмента 3.0, а именно как заряжать энергией людей. В первой из этих глав излагаются теоретические аспекты проблемы, а во второй – практические. Здесь говорится, что люди – это важнейшая часть организации и что менеджеры должны делать все, что в их силах, чтобы сотрудники проявляли активность, были креативными и мотивированными.

Главы 6 и 7 посвящены второму компоненту модели – расширению полномочий команд и созданию условий для их самоорганизации. Для решения этой задачи необходимы полномочия и доверие. Опять же, в первой из двух глав речь идет в основном о теории, а во второй – о практике.

В главах 8 и 9 объясняется, что такое настройка ограничений. Самоорганизация команд может привести к любым результатам, поэтому людям необходимо дать четкое направление и цель, а также обеспечить их защиту и защиту ресурсов, находящихся в общем пользовании. Это третий компонент модели Менеджмента 3.0.

В главах 10 и 11 обсуждается проблема компетенций, недостаток которых может не позволить командам достичь поставленных целей. Поэтому в зоне ответственности менеджеров должна находиться задача развития необходимых

компетенций и дисциплины. Развитие компетенций – это четвертый компонент модели Менеджмента 3.0.

Главы 12 и 13 посвящены пятому компоненту Менеджмента 3.0 – функционированию команд в контексте организации. Подчеркивается важность выбора правильной социально-сетевой структуры, обеспечивающей беспрепятственный обмен информацией.

Главы 14 и 15 рассматривают процесс «Улучшай все», который будет шестым и последним компонентом модели. Подчеркивается необходимость непрерывного улучшения функционирования людей, команд и организации в целом как способа максимально отодвинуть провал. Как и прежде, материал первой из этих двух глав носит более теоретический, а второй – более прикладной характер.

И наконец, в главе 16 мы подводим итоги и проводим общий обзор Менеджмента 3.0, а также его сравнение с некоторыми другими моделями менеджмента.

Как видите, каждый из шести компонентов модели Менеджмента 3.0 описывается двумя главами, и в каждом случае первая из двух глав носит более теоретический, а вторая – более прикладной характер. Можно прочитать только практические главы, чтобы узнать о том, как именно применять гибкий менеджмент, но в этом случае вы не поймете, почему рекомендуются именно эти методы.

Содержание отдельных глав не слишком сильно зависит

друг от друга. Так что *в теории* вы можете читать о шести компонентах модели в любом порядке. Однако *на практике*, вероятно, легче всего начинать с первой главы. Я лично не проверял, какое именно из 720 возможных сочетаний этих компонентов будет наиболее удобным с точки зрения восприятия.

Возможно, время от времени вы будете замечать, что темы, обсуждаемые в этой книге, не всегда тесно увязаны друг с другом. Это сделано намеренно. Мне представлялось важным, чтобы материал был организован вокруг шести компонентов моей модели и чтобы через всю книгу проходило четкое разделение между теорией и практикой. Иногда было нелегко организовать материал в рамках одной главы и четко обозначить связи между различными темами. Но мне кажется, что я справился с этим достаточно хорошо. Выражаю надежду, что восприятие моей книги читателями будет менее критичным, чем восприятие автора.

Содержание книги

Текст книги написан в бета-версии Microsoft Word 2010. Все иллюстрации нарисованы и отсканированы мной, а затем раскрашены в приложении Paint.NET. Иногда в книге попадаются серые вставки с вопросами или замечаниями и ответами на них. Большинство этих вопросов задавались читателями моего блога и рецензентами первых версий книги. Я также использую много ссылок на сайт «Википедии». Некоторые считают, что ссылаться на «Википедию» – порочная практика, но я с этим не согласен. Я предпочитаю ссылки на живой ресурс, над которым постоянно ведется работа по улучшению, чем на ветки потенциально мертвого дерева.

Предвидя обвинения в излишнем теоретизировании, я сделал так, что в сумме объем практических глав превышает объем тех, что отданы теории. Более того, в конце каждой главы есть раздел «Подумать и сделать», что делает книгу еще более полезной с практической точки зрения.

Часто говорят, что применение метафор улучшает понимание абстрактных понятий – именно поэтому я так часто ими пользуюсь. В этой книге вы найдете сравнения менеджеров с садовниками, волшебниками, регулировщиками дорожного движения и другими интересными людьми. Вначале я подумывал назвать эту книгу «Абстрактный садовник». Но в конечном итоге решил использовать другое название,

потому что любая метафора имеет свойство изнашиваться, если ей пользоваться часто и для описания разных явлений. Поэтому теперь при необходимости я стараюсь подобрать отдельную метафору для каждого случая.

У этой книги есть сопутствующий сайт https://www.management30.com. На нем вы найдете дополнительные материалы, не вошедшие в книгу, первоначальные версии иллюстраций (разрешается их похитить и использовать для своих целей), материалы, присланные читателями, и ссылки на другие ресурсы, посвященные гибким методологиям, разработке ПО и теории сложности.

О названии модели

«Менеджмент 3.0» – довольно странное название. Но, как мне кажется, указание на версию 3.0 дает верное представление о направлении развития менеджмента в XXI веке.

Менеджмент 1.0 = иерархии

Некоторые называют этот менеджмент научным, другие – командно-контролирующим. Но в основе одна и та же идея: организацию выстраивают и управляют ею сверху вниз и властные полномочия в руках немногих. У тех, кто находится вверху иерархической организации, самые высокие зарплаты, самые большие эго и самые дорогие офисные кресла. У тех, кто внизу, денег существенно меньше, меньше ответственности и нет мотивации работать хорошо.

В качестве компенсации тех рисков, что несут с собой высокие менеджерские должности, топ-менеджеры наделены возможностью манипулировать бонусами, что во многих случаях позитивно влияет скорее на их личное благосостояние, чем на результаты возглавляемых ими компаний. В качестве побочного фактора извращенные бонусные системы внесли свой вклад в мировой финансовый кризис.

Можно сделать уверенный вывод, что менеджмент 1.0, пусть он до сих пор и наиболее распространенная версия во

всем мире, имеет целый ряд серьезных недостатков. Он устарел и нуждается в обновлении.

Менеджмент 2.0 = дань преходящим увлечениям

Некоторые люди осознают, что вне совсем уж стандартных ситуаций менеджмент 1.0 работает плохо, поэтому были созданы разнообразные и не до конца научные модели и расширения типа системы сбалансированных показателей, шести сигм, теории ограничений и тотального управления качеством. Будучи надстройками менеджмента 1.0, эти модели исходят из того, что организации управляются сверху и призваны помочь топ-менеджерам улучшить «дизайн» своих организаций. Иногда это срабатывает, иногда – нет.

Параллельно возникают другие модели и сервисы, фокусирующиеся на искусстве и мастерстве менеджмента. Многие книги типа «Менеджер за одну минуту», «Двадцать один закон лидерства» или «От хорошего к великому» содержат базовые принципы и рекомендации, которым менеджерам полагается следовать, а также советы больше практиковаться и набирать опыт. Опять же, иногда такие советы и рекомендации работают, иногда – нет. Наборы рекомендаций меняются чаще, чем подгузники у младенца.

Менеджмент 2.0 – это все тот же знакомый нам менеджмент 1.0, к которому добавили некоторое количество над-

строек, чтобы несколько снять остроту проблем, порожденных старомодной системой. Но в основе архитектуры менеджмента 2.0 лежат те же устаревшие иерархии.

Менеджмент 3.0 = сложные системы

В последние несколько десятилетий мы стали свидетелями зарождения и развития теории сложности, вначале в применении к математике и биологии, а затем к экономике и социологии. Это было крупным прорывом. Стивен Хокинг считал это направление в науке настолько важным, что называл XXI век веком сложности.

Одно из важнейших прозрений новой теории заключается в том, что все организации представляют собой сети. Люди могут сколько угодно изображать свои компании в виде иерархий, но это не отменяет того факта, что на практике они будут сетями. Во-вторых, теория сложности в применении к социальным системам показывает, что менеджмент в первую очередь должен заниматься людьми и их взаимоотношениями, а не структурой департаментов и получением прибыли.

Многие из нас уже в курсе, что термин «лидерство» – не более чем модное название ситуации, когда менеджеры делают правильные вещи и делают их правильно. Но мышление категориями сложных систем добавляет в наш словарь новое измерение. Оно заставляет нас воспринимать органи-

зации как живые системы, а не как машины.

Иногда стоит менять названия. От них многое зависит. Название «Менеджмент 3.0» подчеркивает, что менеджмент нуждается в изменениях. Компании Microsoft обычно требуется сделать три релиза продукта, чтобы он нормально заработал. Я считаю, что в своем третьем воплощении менеджмент наконец нашел надежную научную основу. Предлагавшиеся ранее надстройки и апгрейды все еще полезны. Но мы обязаны поменять свою исходную гипотезу с иерархий на сетевые структуры, потому что XXI век – это эпоха сложности.

О подзаголовке книги

В подзаголовке книги «Лидерство и управление командами» упомянута тема **лидерства** – это термин, который часто используется неправильно. Есть два типа людей, которые неверно его интерпретируют. Я называю их принцами и жрецами.

Принцы

Некоторые утверждают, что «лидерство – это не то же самое, что менеджмент» в том смысле, что лидерство предполагает вдохновение, в то время как менеджмент относится скорее к исполнению. Они утверждают, что лидерство находится на «более высоком уровне», чем менеджмент. Меня всякий раз коробит, когда компания называет своих топ-менеджеров «лидерами».

Каждый сотрудник, начиная от президента компании и вплоть до последнего разработчика, может вдохновлять коллег и указывать им направление. У лидеров по определению нет формальных рычагов власти над своими последователями. Но какой же акционер доверит деньги «лидеру», не располагающему формальными полномочиями? Это глупая затея.

К сожалению, в данный момент среди топ-менеджеров

распространена мода называть себя лидерами независимо от того, есть у них последователи или нет. Топ-менеджеры используют «лидерство» как социальный миф для укрепления своих позиций в организационной иерархии [Hazy 2007: 110]. Я называю таких топ-менеджеров *принцами* (и принцессами), поскольку они думают, что занимаемая должность дает им больше прав на роль лидера, чем всем остальным, а еще потому, что они предпочитают блестящие предметы здравому смыслу.

Жрецы

Еще одна категория людей утверждает, что «менеджмент не нужен». Они говорят о социальных сетях, «Википедии», Linux и других замечательных достижениях социальных групп, которым удалось сформировать общую цель и в результате многого добиться. Они полагают, что «самоорганизующиеся» группы вообще не нуждаются в менеджерах, только в лидерах, обладающих видением.

К сожалению, данная точка зрения игнорирует тот факт, что ни в одном из таких примеров речь не идет о бизнесе. Если никто не будет *владельцем* активов организации, то никто и не нужен, чтобы *управлять* ими. Акционеры вряд ли оценят, если в результате самоорганизации их биотехнологическая компания будет трансформирована в кейтеринговый бизнес. Не стоит принимать во внимание и мнение сотрудни-

ков о том, нужны им менеджеры или нет. Менеджеры необходимы акционерам для того, чтобы управлять их бизнесом. Самоорганизация сама по себе лишена ценности. Требуется кто-то заинтересованный в результате, кто и будет решать, «хороши» или «плохи» результаты самоорганизации.

Увы, некоторые считают, что иерархия – это всегда «плохо», а самоорганизация – всегда «хорошо». Я называю их жрецами (и жрицами), потому что они проповедуют свою веру в то, что считают «хорошим», хотя (как показано в этой книге) никаких научных оснований для этой веры нет.

Прагматики

Когда речь идет о менеджменте и лидерстве, реальность требует от нас оставаться прагматиками. Любой бизнес нуждается в менеджменте от лица акционеров. Да, у менеджеров *должны* быть лидерские качества. Но многие лидерские роли могут исполняться самоорганизующимися людьми (не занимающими менеджерских должностей), находящимися на самых разных позициях в компании. Эти неформальные лидеры должны понимать, что направление, в котором происходит самоорганизация, необходимо немного корректировать и что делается это акционерами через распределение полномочий среди менеджеров.

Если вы похожи на меня, то вы не принц и не жрец, вы – один из простолюдинов. Я буду называть нас прагматика-

ми. Мы понимаем, что управленческая иерархия – это базовая необходимость (и нечем тут хвастаться) и что основная часть работы совершается внутри социально-сетевой структуры, состоящей из равных: лидеров и последователей. Коммуникация осуществляется через сети, а полномочия – через иерархию.

Я написал эту книгу для прагматиков.

Глава 1
Почему все не так просто

Всякая сложная проблема имеет решение – простое, удобное и ошибочное.
Г.Л. Менкен, журналист и писатель (1880–1956)

Однажды мне удалось некоторое время побыть миллионером – по крайней мере на бумаге. Частные инвесторы оценили мой стартап в интернете в 10 миллионов евро, при этом мне принадлежало 70 % финансовой фикции, которую им удалось создать вокруг моего проекта. Мне даже присудили титул *предпринимателя года*, поскольку я очень хорошо передал свое видение проекта. А созданные мной цветные графики, показывающие ожидаемые доходы и расходы, выглядели просто сказочно – опять же на бумаге.

Несмотря на все это, деньги, которые мы с инвесторами вложили в проект, не привели к росту прибыли. Создание дополнительного контента не привело к увеличению посещаемости нашего сайта. Мы наняли больше программистов, но это не сказалось каким-либо существенным образом на скорости разработки. Договоренности, которые у нас были с другими сайтами, не привели к росту доходов. Доходы оказались даже *ниже*, чем перед первым раундом инвестиций. Уверен, что вам незнакомо название нашего не слишком из-

вестного сайта. Все попытки раскрутить его позорно провалились. А когда лопнул пузырь доткомов, это вообще поставило жирный крест на всем проекте – так же, как и на множестве других, которыми были заняты все вокруг нас.

И тем не менее сам процесс был весьма увлекательным. Мы многому научились. О, как многому мы научились! Если правда, что лучше всего учишься на собственных ошибках, то к настоящему моменту я должен был быть просто всеведущим божеством. В качестве менеджера, руководившего процессом разработки, лидера команды, менеджера проекта и просто разработчика программного обеспечения я совершил столько ошибок, что мне до сих пор кажется странным, что вместе со своим проектом я не обрушил весь интернет. Но мы действительно в результате очень многому научились.

При написании этой книги меня поддерживала надежда, что и вы многому научитесь на моих ошибках и ошибках тех, кто совершал их до меня. За последние десять лет я понял, что при разработке программного обеспечения оптимальные результаты дают именно **Agile-методологии** (см. главу 2 «Гибкие методологии разработки ПО»). Я также убедился, что серьезнейшее препятствие на пути принятия Agile-методологий во всем мире – это традиционный менеджмент. Я исхожу из представления, что вы либо руководитель, либо просто интересуетесь менеджментом. Возможно, вы разработчик, технический директор, глава проектной группы или тестировщик. На данный момент это не имеет особого зна-

чения. Важно то, что вы хотите больше узнать о менеджменте – о так называемых Agile-методологиях менеджмента и разработки. Обещаю, что вы действительно узнаете много нового. Задача этой книги – научить вас хорошо разбираться в *гибких* методологиях и помочь в создании *гибкой* организации. Мы скоро перейдем к конкретному обсуждению гибких методологий, но сначала необходимо уделить внимание основам, которые заключаются в знании того, как ведут себя люди и системы. Вы спросите: «Зачем это нужно?» По той же причине, по которой врачи сначала изучают, как устроен человеческий организм, пилоты постигают функционирование самолета, а программисты знакомятся с устройством компьютера. Ну а менеджеры должны знать, как функционируют социальные системы.

На *своем* горьком опыте я убедился, что, как бы детально мы ни планировали тот или иной проект, в реальности события почти наверняка будут развиваться по-другому. Системе, в которой вы функционируете, безразлично, какие у вас планы. Возможно, вы полагаете, что из точки A можно попасть в точку B, и при этом не исключено, что вы правы – но только в теории. Но теории редко срабатывают на практике, а у *предсказуемости* есть коварная сестра, которую зовут *сложность*.

Но я забегаю вперед. Как будет показано далее, люди предпочитают *воспринимать* происходящее линейно, а следовательно, скорее всего, я поступаю правильно, линейно

выстраивая изложение в книге. Эта история берет свое начало в причинно-следственных связях. В данной главе мы исследуем эти связи и нелинейность, а ближе к концу главы познакомимся с моделью Менеджмента 3.0.

Причинно-следственные связи

Представлением о том, что обычно все происходит в соответствии с нашими планами (как казалось и мне, когда я был миллионером на бумаге), мы обязаны своей врожденной склонности к **детерминизму**. Он утверждает, что «будущие события неизбежно вытекают из предыдущих в соответствии с законами природы». Детерминизм говорит нам, что причиной любого события является другое событие, произошедшее ранее. С логической точки зрения это значит, что если нам известно все о текущем состоянии дел и мы знаем все варианты перехода из одного состояния в другое, то мы должны быть способны предсказывать будущие события, рассчитав их на основе предшествующих событий и законов природы. Если вам кинуть мяч, вы можете поймать его, поскольку в состоянии определить его траекторию. Вы вполне способны оценить, сколько у вас останется денег до конца месяца после того, как вы хорошенько погуляете с друзьями; или как лучше вывести из себя брата либо сестру и при этом не получить по шее.

В мире науки детерминизм оказался чрезвычайно успешным, позволив ученым предсказывать огромное количество разнообразных событий и явлений. Например, используя механику Ньютона, ученые уверенно предсказывают возвращение кометы Галлея в Солнечную систему в 2061 году. Науч-

ный метод предсказания будущих событий на основе событий, им предшествовавших, а также законов природы оказался настолько успешным, что философ Иммануил Кант провозгласил всеобщий детерминизм в качестве необходимого условия любого научного знания [Prigogine, Stengers 1997: 4].

Детерминизм позволяет разработчикам программного обеспечения проектировать, планировать и предсказывать поведение своего программного продукта в реальных условиях использования. Они создают или вносят изменения в программный код, чтобы задать или изменить поведение программного продукта после компиляции и развертывания у пользователя. Если на мгновение отвлечься от ошибок программирования, сбоев операционных систем, аварийных отключений электричества, неквалифицированных пользователей и других рисков, то можно сказать, что предсказания разработчиков очень часто оказываются весьма точными. Тот же детерминизм позволил мне в свое время сделать вполне верный прогноз, что мой проект обанкротится, если не удастся найти больше клиентов.

Но, как это ни было бы странно, одного детерминизма недостаточно. Хотя мы умеем предвидеть очередное появление кометы Галлея и можем еще на стадии разработки предсказать, как будет функционировать программное обеспечение, мы не в состоянии определить погоду на месяц вперед. Мы также не в состоянии точно предвидеть результат слож-

ной комбинации желаемых параметров программного обеспечения, время, имеющееся на разработку, требуемые для проекта ресурсы или (что, к сожалению, случилось с моим проектом) наступление момента, когда появятся новые клиенты.

Так в чем же проблема?

Сложность

Если вежливый и послушный сын ваших соседей – олицетворение предсказуемости, то его своенравная и взбалмошная младшая сестра может служить символом **сложности**. Предсказуемость позволяет вам ходить на работу, назначать встречи, заниматься спортом и смотреть телевизор. В то же самое время сложность зачастую превращает взаимодействия между вами и внешним миром в непредсказуемый хаос, полный неожиданных проблем и сюрпризов.

Многие иногда путают создаваемые сложностью проблемы с проблемой больших чисел (когда одновременно происходит огромное количество событий), но сложные явления не всегда предполагают наличие большого количества элементов. Возьмем, например, молекулу воды (фигурально выражаясь, естественно, на практике это сделать очень непросто). Эта молекула состоит всего из двух атомов водорода и одного атома кислорода. Ничего сложного, не так ли? И тем не менее даже такая простая структура из трех атомов проявляется неожиданным образом в сложных явлениях текучести, эффектах, связанных с плотностью воды, и других физических и химических явлениях [Solé 2000: 13], которые не поддаются легкому объяснению с точки зрения поведения отдельных атомов. Таким образом, сложность необязательно будет проявлением больших чисел. Достаточно трех мо-

лекул воды, чтобы состоящая из них система характеризовалась сложным поведением – примером будет знаменитая **задача трех тел**.

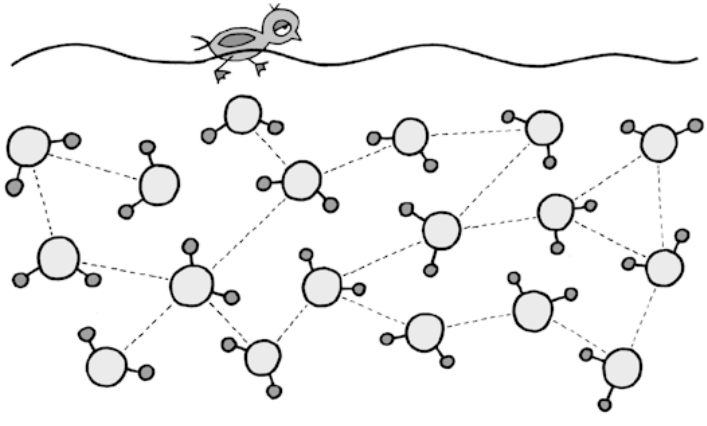


Рис. 1.1. Что же на самом деле происходит в воде?

К счастью, с того момента, когда Кант с энтузиазмом объявил причинность основой научного знания, наука не стояла на месте. Теория динамических систем, теория хаоса, теория сетей, теория игр и ряд других научных дисциплин добились значительного прогресса, объяснив, *почему* некоторые явления невозможно предсказывать и почему некоторые события невозможно планировать или рассчитать заранее – их можно только испытывать или наблюдать. Часто весь комплекс исследований в области сложных систем собирательно имену-

ют **теорией сложности** (см. главу 3 «Теория сложности»).

Если развитие науки начиная с XVII века проходило под знаком детерминизма, то сложность как предмет исследования возникла в XX веке; соответствующие исследования значительно ускорились с того момента, когда в конце XX века **теория сложности** выделилась в отдельную научную дисциплину. Физик-теоретик Стивен Хокинг утверждал, что XXI век будет веком сложности [Chui 2000].

Развитие теории сложности – хорошая новость для руководителей, лидеров команд и менеджеров проектов (а также всех прочих «лидеров» и «менеджеров»), работающих в компаниях, создающих ПО. Это означает, что возник *научный* подход к исследованию сложных систем, включая проблемы разработки программного обеспечения и управления организациями в целом. И хотя для меня момент истины опоздал ровно на 10 миллионов евро, я согласен со Стивеном Хокингом, что представление о сложности – ключевая парадигма XXI века.

Наше линейное мышление

К сожалению, применяя теорию сложности к решению конкретных проблем, мы постоянно сталкиваемся с определенным неудобством: наш мозг предпочитает видеть простые причинно-следственные связи и игнорировать сложность. В своей статье «Рожденные верить: Как наш мозг создает богов» [Brooks 2009] автор показывает, что человеческий мозг чрезмерно ориентирован на установление причинно-следственных связей, что заставляет нас видеть их даже там, где их нет. Как отмечается в статье, дети думают, что острые скалы созданы для того, чтобы о них могли чесаться животные, а реки – чтобы по ним можно было плавать на лодках. По всей видимости, устройство человеческого мозга заставляет нас повсюду усматривать целеполагание и причинно-следственные связи, даже если для этого нет никаких оснований.

> «Вы слышите шорох в кустах и делаете вывод, что там кто-то притаился». ‹…› Вероятно, чрезмерная склонность повсюду усматривать причины и следствия дает преимущество с точки зрения выживания. Если вокруг полно хищников, недостаточно обнаруживать их в девяти случаях из десяти. Лучше перестраховаться и убежать, даже если в некоторых случаях опасность окажется иллюзорной. В конце концов, цена такой

перестраховки невелика[1].

Наш мозг жестко запрограммирован отдавать предпочтение «линейному мышлению» (представлению о предсказуемости следствий, если известны причины) перед «нелинейным» (гипотезой, что в реальности все обстоит гораздо сложнее). Мы привыкли считать, что события от начала и до конца разворачиваются линейно. В школе нас учат решать линейные уравнения, а более часто встречающиеся на практике нелинейные игнорируются просто потому, что справиться с ними гораздо труднее. Нам легче принять утверждение «это сделал он», чем утверждение «некоторые вещи просто случаются». Если в наличии проблема B, то мы предполагаем, что ее причиной стало событие A. Причиной финансового кризиса стали банкиры; в сокращении числа рабочих мест виноваты иммигранты; в плохой атмосфере в компании виноват менеджер; таяние полярных льдов вызвано выбросами CO2; проектной группе не удалось уложиться в отведенные сроки из-за того, что кто-то плохо работал. Линейное мышление воспринимает мир как пространство, наполненное легкообъяснимыми событиями, вызванными простыми причинами и имеющими простые следствия. Джеральд Вайнберг называет это **ошибкой причинности** [Weinberg 1992: 90].

Наша мыслительная зависимость от детерминизма застав-

[1] Michael Brooks, "Born believers: How your brain creates God," New Scientist, February 4, 2009 [Brooks 2009: 32].

ляет людей искать способы *контроля*, которые позволили бы обеспечить наступление *желательных* событий и ненаступление *нежелательных*. В конце концов, если известно, что ситуация А имеет своим результатом событие В, а ситуация А′ – событие С, при этом С лучше, чем В, то всего-навсего надо заставить А превратиться в А′, и все будет хорошо. Так по крайней мере часто кажется.

Инженеры и другие люди с техническим складом ума особенно восприимчивы к идеям, базирующимся на идее управляемости. Именно инженеры создали **научный менеджмент**, основанный на отдаче распоряжений и контроле их исполнения, который всецело господствовал с начала XX века. И именно они придумали системы контроля, которые до сих пор существуют во многих организациях [Stacey 2000a: 7]. Сейчас уже всем известно, что системы контроля эффективны, только если речь идет о повторяющихся операциях, не требующих особых размышлений. Но они *не работают* в ситуациях, когда необходим творческий подход при разработке новых продуктов! Поэтому было бы только справедливо, если бы инженеры и вытащили нас из того управленческого болота, в которое они нас в свое время затянули.

Детерминизм в управлении побуждает менеджеров искать причины, которые привели бы к получению точно такого результата, который им необходим, путем предварительного проектирования этого результата и детального планирования сверху вниз. Чем крупнее организация, тем больше уси-

лий затрачивается на анализ поведения ее составных частей с целью заставить их взаимодействовать так, чтобы в результате поставленные цели были достигнуты.

Прежде и я охотно создавал себе мир иллюзий, основанный на предварительном проектировании и планировании сверху вниз. Мой бизнес-план (который, кстати, был даже отмечен профессиональными премиями) состоял из 30 страниц тщательно выверенной чепухи. В нем в деталях было расписано, как мы разбогатеем. В тот момент мы верили в этот бизнес-план. В конце концов, поскольку его разработал я, как он мог быть неправильным?

Редукционизм

Подход, в рамках которого систему разбирают на части, а затем изучают взаимодействие этих частей, чтобы понять, как работает целое, называется **редукционизмом**. Суть подхода в том, что «явления могут быть исчерпывающе объяснены в терминах других, более фундаментальных явлений». Мы можем разобрать самолет на детали и увидеть, как он функционирует, изучив каждый винтик; мы можем понять, как работает компьютерная программа, проанализировав ее код; в настоящее время ученые пытаются изучать болезни и врожденные дефекты, анализируя геном человека в надежде идентифицировать отдельные гены, ответственные за те или иные проблемы.

Редукционистский подход хорош только до определенного предела (рис. 1.2). После многолетних исследований ученые все еще не понимают, как работает сознание. Несмотря на созданные за последние сто с лишним лет многочисленные теории, экономисты так и не смогли предложить модель, которая позволяла бы достоверно предсказывать экономические кризисы. Многочисленные теории изменения климата дают совершенно разные прогнозы последствий глобального потепления. И хотя нет недостатка в методиках, моделирующих процесс разработки программного обеспечения, тем не менее во всем мире множество проектов все еще натал-

киваются на непредвиденные проблемы. Живые организмы, сознание человека, экономика, изменение климата и проекты по разработке ПО – все эти системы устроены таким образом, что их поведение невозможно предсказать путем деконструкции и изучения компонентов по отдельности.

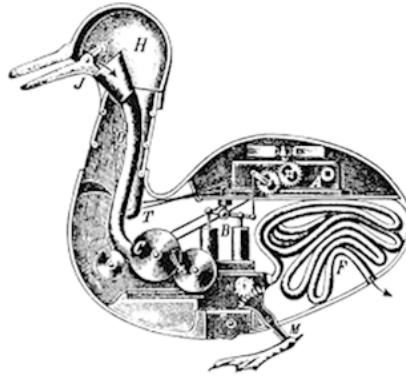


Рис. 1.2. Пример редукционизма, зашедшего слишком далеко

Примечание: Рисунок из «Википедии»,находится в свободном доступе.

Способность людей интерпретировать окружающую действительность весьма ограниченна

В процессе подготовки этой книги некоторые рецензенты указали мне, что отличительная черта людей – тенденция неточно интерпретировать

информацию, поступающую из окружающей действительности. Мы склонны игнорировать любые факты, в которые мы не верим и которые противоречат уже сложившимся в наших головах моделям. Такая избирательность действительно мешает нам более или менее точно предвидеть будущие события.

Идея целостности

Холизм как теория предполагает, что поведение системы несводимо к сумме поведений ее отдельных частей, а напротив, решающим образом определяется ее свойствами как единого целого. Этот подход часто воспринимают как противоположность редукционизму, хотя ученые, исследующие такие системы, полагают, что сложность будет связующим звеном между обоими подходами и каждый из них необходим, но недостаточен [Corning 2002: 69].

Даже некоторые из наиболее рьяных редукционистов отказались от представления, что все явления могут быть сведены к поведению их составных частей. Философ Дэниел Деннет предложил термин **«Жадный редукционизм»** [Dennet 1995] для обозначения форм редукционизма, которые стремятся *полностью* вывести поведение системы из взаимодействия между ее составными частями. Например, утверждение, что «гиперссылки – это не более чем электроны и на самом деле они не существуют» будет примером такого жадного редукционизма. В качестве контраргумента я сказал бы, что если жадные редукционисты правы, то значит, они тоже не существуют, и это полностью снимает все их смехотворные аргументы. Но я отвлекся.

В качестве компромисса биолог-эволюционист Ричард Докинз предложил понятие **иерархического редукцио-**

низма [Dawkins 1996], смысл которого в том, что сложная система может быть представлена в виде иерархии, в которой события на каждом уровне могут быть объяснены поведением компонентов, находящихся в данной иерархии одним уровнем ниже, но только одним уровнем. Если следовать этой логике, вы не сможете объяснить провал своего проекта тем, что вам помешали кварки и лептоны.

Многие ошибочно полагают, будто бы из редукционизма следует, что мы в состоянии реконструировать любую систему, если понимаем, как функционируют ее составные части. В этом и состоит заблуждение: даже если мы отлично понимаем, как ведут себя все компоненты системы, это не значит, что система сводится к сумме своих составных частей [Miller, Page 2007: 41]. Знание компонентов, находящихся на нижних уровнях системы, вовсе не означает, что мы сможем воссоздать всю систему как единое целое. Интересно, что, если исходить из редукционистского подхода и отследить изначальную причину проблемы (например, воспользовавшись *методикой анализа основной причины*), мы все равно не сможем создать систему, в которой данная проблема отсутствовала бы. Например, мы можем установить причину конкретного случая сердечной недостаточности (редукционизм), но нам никогда не удастся создать сердце, которое принципиально не будет подвержено сердечной недостаточности (конструкционизм).

Значит ли все это, что методика анализа основной причины бесполезна?

Нет, она *чрезвычайно* полезна. Говоря о ее недостатках, мы лишь имеем в виду, что она обращена исключительно в прошлое. С ее помощью можно решить проблемы, которые уже случились, и исключить их возникновение в будущем. Но эта методика никак не помогает предсказывать, *что именно в будущем может пойти не так*.

Иерархический менеджмент

Взгляд на систему как на единое целое (холизм) и иерархический редукционизм сходятся на том, что не все в поведении сложной системы может быть объяснено событиями, происходящими на ее более низких уровнях. Обе гипотезы допускают, что каждому уровню присущи свои особенные и не сводимые к более элементарным уровням свойства. Например, как бы тщательно вы ни вглядывались, у вас не получится без определенных затруднений идентифицировать внутри деконструированной утки рычажки, подшипники и шестеренки, предназначенные для ходьбы, плавания и кряканья (см. рис. 1.2). И тем не менее, увидев этот объект в парке, вы сразу понимаете, что это утка.

Все вышесказанное имеет далекоидущие последствия для менеджеров сложных систем вроде нас с вами, а также для менеджеров, управляющих разработками, проектных менеджеров и лидеров команд. Это означает, что тот, кто знает все о функционировании определенного уровня в иерархической системе, может оказаться недостаточно квалифицированным, чтобы работать на других уровнях в той же системе, потому что для этого требуются иные знания. Молекулярный биолог может оказаться недостаточно «квалифицированным», чтобы выполнять обязанности садовника, потому что знание того, как функционируют живые организ-

мы на уровне клеток-эукариотов, генов и РНК, не подразумевает умения ухаживать за садом; а хорошему садовнику совсем необязательно знать о хромосомах и геноме. Точно так же генеральный директор должен иметь обширные знания о том, как управлять компанией, но при этом он может быть полным профаном в том, что касается коучинга и других навыков управления людьми (я уверен, что многие читатели лично сталкивались с такими ситуациями).

Управление организацией требует совершенно иных знаний и опыта, чем управление людьми, хотя *некоторое* представление о том, как система функционирует на более низких уровнях, *может* оказаться полезным. Инженер-программист Джоэл Сполски предложил **закон дырявых абстракций** [Spolsky 2002] в качестве объяснения, почему в системах компоненты, находящиеся на более высоких уровнях, могут проявлять себя неожиданным образом в результате воздействия на них событий, происходящих на более низких уровнях, хотя более высокие уровни, по идее, должны быть изолированы от такого воздействия. Более высокие программные уровни, которые подвергаются воздействию событий, происходящих на более низких программных уровнях, считаются дырявыми. Типичным свидетельством такого рода дырявых абстракций в программировании будут непонятные сообщения об ошибках, которые получают пользователи (рис. 1.3).

Рис. 1.3. Проявление дырявой абстракции?

Мы наблюдаем аналогичные проблемы в других сложных системах. Мое сознание нередко становится жертвой временных помутнений, дежавю, забывчивости, произвольно всплывающих воспоминаний и иных странных вещей, которые могут быть объяснены только как сбои в нейронных цепочках низшего уровня, проникающие на более высокий уровень, который я называю своим мышлением. И тем не менее мне вовсе не нужно анализировать нейронные цепочки, чтобы обеспечить удовлетворительную работу своего сознания, хотя порой приятно осознавать, что, по мнению неврологов, такого рода явления достаточно распространены. Точно так же вам не нужно хорошо разбираться в программировании на языке ассемблера, чтобы создавать хорошие программы более высокого уровня, хотя опять же *некоторое* представление о том, как это все работает уровнем ниже, может иногда облегчить вашу жизнь. В менеджменте все то же самое. Чтобы управлять компанией, генеральному директору необязательно уметь эффективно общаться с людьми, при

условии что коммуникация делегирована надежной команде других управленцев (в этом его отличие от менеджеров по разработке новых продуктов, проектных менеджеров и лидеров команд, которым необходимо ежедневно общаться с коллегами). Но на случай, если проблемы нижнего уровня все же прорываются на более высокий (иными словами, когда имеет место дырявая абстракция), владение некоторыми навыками коммуникации может пригодиться.

Гибкий менеджмент

Когда иерархический менеджмент встречается со сложными системами и нелинейным мышлением, мы попадаем в область, которую я называю **гибким (Agile) менеджментом**. Это логическое дополнение к **Agile-методологиям разработки программного обеспечения**, которые были созданы в 1990-х годах несколькими группами и отдельными специалистами. Необходимость нового подхода была продиктована неудачами при разработке программного обеспечения, к которым приводил детерминистский подход, основанный на тщательном контроле, предварительном детальном проектировании и планировании сверху вниз. Несмотря на весь этот интенсивный менеджмент, результатом во многих случаях было программное обеспечение, работавшее из рук вон плохо.

Гибкие методы разработки ПО некоторыми своими корнями уходят в теорию сложности, признающую недостаточность причинного детерминизма для реализации успешных проектов. Такие хорошо известные и используемые в гибких методологиях понятия, как *«самоорганизация»* и *«эмерджентность»,* напрямую взяты из литературы по сложным системам [Schwaber, Beedle 2002], и люди, практикующие в настоящее время Agile-методологии, понимают, что при конструктивистском подходе гарантированы неудачи. Толь-

ко непрерывная идентификация возникающих в ходе проекта проблем и устранение их причин позволяют последовательно развивать проект по разработке ПО и в конечном итоге получить на выходе успешный программный продукт. Это похоже на процесс взросления или воспитания детей.

Несмотря на блестящие успехи с точки зрения окупаемости инвестиций Agile-проектов [Rico 2009], многие менеджеры по всему миру в своих компаниях препятствуют гибкому проектному менеджменту и гибким методологиям. Исследования и опросы свидетельствуют, что основными препятствиями на пути принятия гибких методов разработки ПО становятся традиционные методы управления изменениями, организационная культура, недостаток поддержки со стороны руководства, низкая подготовленность персонала, а также внешнее давление [VersionOne 2009]. За многое из этого отвечают именно менеджеры. Если верить имеющимся отчетам на эту тему (а у меня нет оснований в них сомневаться), то сами менеджеры во всем мире будут скорее проблемой, чем частью решения. Печально, что это характерно не только в случаях внедрения гибких методологий разработки ПО. То же самое происходит при любых других серьезных организационных изменениях.

Моя позиция в этой книге состоит в том, что, когда необходимы подобные изменения, традиционный менеджмент будет проблемой, а не решением. Эту же точку зрения много лет назад высказывал Уильям Эдвардс Деминг. Именно по-

этому нам нужна теория гибкого менеджмента: теория, которая хорошо сочеталась бы с гибкими методологиями разработки ПО.

Моя теория всего

Существует ли теория, которая может помочь менеджерам работать в гибкой среде? За последние несколько десятилетий было предложено много управленческих теорий – впрочем, большинство из них вовсе не будут теориями в строгом научном смысле [Lewin, Regine 2001: 5]. Настоящая научная теория должна быть в состоянии не только указывать на существование каких-либо природных явлений, но и делать проверяемые утверждения о наблюдаемом реальном мире, предсказывая, каких событий следует ожидать, *прежде* чем их можно будет наблюдать. Именно в этом смысле различные управленческие «теории» не соответствуют ожиданиям. Они зачастую представляют собой не столько теории, сколько наборы технических приемов. Вместо того чтобы давать описание того, как функционирует реальный мир, они предлагают советы (часто полезные), как подойти к той или иной проблеме или ситуации. В этом смысле хорошим примером будет теория ограничений. Это не научная теория, а управленческая философия, которая предлагает способы оптимизации процессов и позволяет добиваться целей, постоянно фокусируясь на имеющихся ограничениях.

Значит ли это, что я в состоянии предложить свою собственную «теорию» гибкого менеджмента, втайне надеясь войти в пантеон таких мыслителей, как Портер, Деминг и

Друкер? Боюсь, что нет.

Было время, когда я надеялся изобрести *теорию всего*, что касалось бы управления разработкой программного обеспечения. Эта теория описывала бы универсальные принципы функционирования любых команд, работающих над созданием программных продуктов, и предоставила бы в помощь менеджерам единую и законченную модель управления такими командами. Оглядываясь назад, я думаю, что в то время мое мышление было жертвой дырявой абстракции.

К счастью, я вскоре понял, что эта цель недостижима по двум причинам. Во-первых, уже существует множество теорий, претендующих на объяснение того, как люди работают в командах (здесь можно порекомендовать книгу «Малые группы как сложные системы» (Small Groops as Complex Sistems) [Arrow 2000], а также журнал «Эмерджентность. Теория сложности в применении к организациям»[2]). Эта область известна как **социальная сложность**. Во-вторых, сама теория сложности говорит нам, что любые попытки создать единую модель, описывающую сложные системы, неизбежно обречены на неудачу. Эта тема затронута в главе 16 «Все модели неверны, но некоторые из них полезны». Я испытал облегчение, когда понял это: «Сделать это невозможно. Прекрасно! Значит, я могу начать работать над чем-то другим!» Это отличный пример того, когда понимаешь свои

[2] Журнал E: CO. Emergence: Complexity & Organization выходит в издательстве Emergency Publications.

заблуждения еще на раннем этапе. (Из **теорем Гёделя о неполноте** следует, что такая невозможность распространяется и на любые единые теории. Все-таки хорошо, что ученые в своих поисках не сдаются так легко, как я.)

Модель, предлагаемая в этой книге

Эта книга поможет вам стать более хорошим менеджером. В частности, в ней показано, в чем должны заключаться ваши обязанности в качестве Agile-руководителя в компании, занимающейся разработкой программных продуктов на основе гибких методологий. Книга дает богатый инструментарий, что позволяет применять теорию в ежедневной практике. Она объясняет, как нужно управлять командами, исходя из представления, что системы в большинстве случаев оказываются сложными, а не линейными, а также как фокусироваться на адаптивности, а не на предсказуемости. Не имеет особого значения, кто вы – менеджер проекта по разработке ПО, лидер команды, технический директор или программист. В конечном итоге все мы пытаемся управлять окружающей нас средой. Давайте научимся делать это хорошо.

Применяемая в книге модель показана на рис. 1.4. Я называю ее моделью Менеджмента 3.0. Она рассматривает организацию с шести углов зрения. Каждый из этих компонентов описан в книге отдельно, и каждому посвящено по две главы – теоретической и практической. Но прежде чем мы начнем обсуждать модель в деталях, я считаю важным еще раз вернуться к двум базовым комплексам идей, лежащим в ее основе, а именно к *гибкости* и *сложности*, а также уделить немного времени истории каждого из этих комплек-

сов. Глава 2 содержит краткий обзор гибких методологий разработки программного обеспечения, а в главе 3 рассматриваются основы теории сложности. Суть модели, то есть способы управления командами разработчиков, вы найдете в центральной части книги, которая открывается главой 4 «Информационно-инновационная система» и заканчивается главой 15 «Улучшение всего». И наконец, глава 16 содержит краткое заключение.

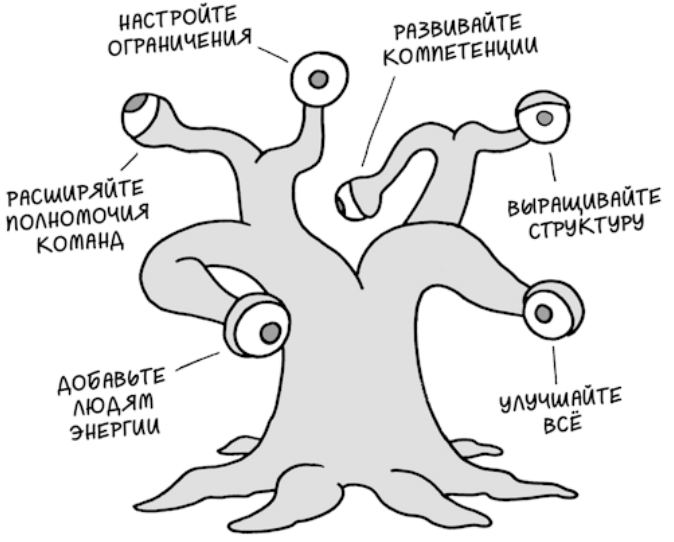


Рис. 1.4. Модель Менеджмента 3.0

Хотелось бы мне, чтобы подобная книга попала мне в ру-

ки десять лет назад, когда я занимался своим стартапом. Но в этом случае вполне *могло случиться*, что я все же заработал бы свои миллионы и, по всей вероятности, вряд ли стал бы заморачиваться написанием этой книги. По-видимому, все это означает, что планирование карьеры зачастую бывает совершенно бесполезно, а неудачный проект может обернуться удачей – надо только суметь вовремя это увидеть.

Резюме

Человеческий мозг устроен таким образом, чтобы за каждым событием видеть определенную причину. Такое стремление к определению причинно-следственных связей полезно при прогнозировании и планировании. Однако очень часто ситуации куда сложнее, чем может показаться на первый взгляд. Теория сложности говорит нам, что применение линейного мышления для решения сложных проблем чревато болезненными ошибками.

Несмотря на то, что редукционизм (понимание природы системы через осмысление ее составных частей) оказался успешным в науке, в настоящее время общепризнанно, что, следуя по этому пути, можно зайти слишком далеко.

Чтобы разобраться во многих сложных проблемах, необходим более целостный подход, применяемый при изучении сложных социальных систем. Такой подход предлагает целостное представление о взаимодействиях, происходящих в группах людей.

Менеджмент 3.0 – это модель для Agile-менеджмента, которая позволяет применять подходы теории сложности к управлению командами, занимающимися разработкой программных продуктов с использованием гибких методологий.

Подумать и сделать

Давайте посмотрим, как применить некоторые идеи данной главы в вашей компании:

• Возьмите в качестве примера какую-нибудь реальную нерешенную проблему из своей профессиональной сферы. Попробуйте представить, в чем может состоять ее причина. Уверены ли вы, что эта причина – единственная? Почему вы так считаете? Обсуждали ли вы эту проблему со всеми заинтересованными сторонами? Все ли из них согласны, что проблема вызвана единственной причиной? Проведите такой же анализ для каждой из наиболее важных проблем, с которыми имеете дело. Убедитесь, что вы не упрощаете сложность данных проблем и не пытаетесь устранить причину, которая определена неправильно.

• Если сотрудники вашей компании при анализе проблем пользуются какими-то методиками анализа основной причины (например, методикой пяти «почему»), попробуйте обсудить с ними встроенную в эти методики тенденцию зачастую неоправданно упрощать отношения причины и следствия. У многих явлений, происходящих в сложных системах, имеются *множественные* причины, а кроме того, причины и следствия могут находиться в отношениях циклической зависимости. В таких ситуациях ни одна из причин не будет главной, поэтому методика анализа основных причин не может

в полной мере отражать сложность окружающего мира. Преодолеть упрощенный взгляд на ситуацию можно с помощью дискуссии на эти темы с компетентными коллегами. Организуйте такое обсуждение.

Глава 2
Гибкие методологии разработки ПО

Каждое утро я встаю, преисполненный решимости изменить мир и прекрасно провести время. Иногда это затрудняет составление плана на день.
Э.Б. Уайт, американский писатель (1899–1985)

Для некоторых из вас эта глава необязательна. Если вы знакомы с Agile-методологиями разработки программного обеспечения[3], вы уже и так много знаете (если не все) о том, о чем тут пойдет речь. Эта глава скорее предназначена для тех читателей, которые хотели бы узнать немного больше об истории и основах гибких методологий *прежде*, чем мы начнем говорить о роли менеджеров в гибких организациях (глава 4 «Информационно-инновационная система»).

В этой книге я исхожу из представления, что вы *знаете* лишь кое-что об основах гибких методологий. А пока сделайте вид, будто считаете, что XP – это старая операционная система, и продолжайте читать.

---

[3] Далее в книге в качестве синонимов будут использоваться термины «гибкие методологии», «Agile-методологии» и производные от них.

Прелюдия к гибким методологиям

Я люблю считать деньги почти так же, как и тратить их. В начале 1990-х годов, когда я учился в Делфтском техническом университете, в свободное время я написал бухгалтерскую программу. Мне было интересно этим заниматься, несмотря на небольшое неудобство: денег, которые нужно было считать, у меня не было. Не исключено, что где-то в глубине души я надеялся, что миллионы появятся автоматически, как только я буду готов их сосчитать. Увы, этого не случилось.

Я был единственным автором этого программного продукта (около 30 000 строк кода). Я не владел формальной методологией, у меня было мало опыта создания ПО, а также не было менеджера, коуча или ментора. Но у меня имелось время, компьютер, видение и страстное желание создать великолепный продукт (рис. 2.1).

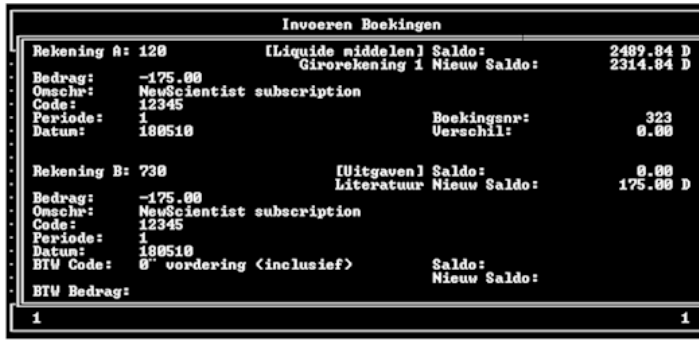


Рис. 2.1. JEBS 2.0, бухгалтерская программа, написанная мной 20 лет назад (на голландском)

К своему удивлению, я сумел продать эту программу дюжине клиентов, и некоторые из них испытали приятный шок от того, что программный продукт может быть простым, удобным для пользователей и иметь симпатичный интерфейс (для программы, написанной в 1990-е годы). Хотя прошло уже двадцать лет, я до сих пор пользуюсь этой программой для управления своими финансами.

Как это вообще возможно? Как неопытному программисту удалось создать продукт столь высокого качества, что он работает почти безупречно вот уже почти двадцать лет?

Не имею ни малейшего представления.

Но… У меня есть список из нескольких пунктов, которые должны быть знакомы последователям гибких методологий разработки.

• **Я работал над своим продуктом с энтузиазмом**. У меня был кое-какой опыт взаимодействия с бухгалтерскими приложениями, и я был убежден, что их разработали в аду с целью лишить пользователей всех жизненных и душевных сил. Мое видение состояло в том, что мой продукт будет совершенно *иным*. В отличие от других программ, имевшихся на рынке, пользоваться моим продуктом будет одно удовольствие.

• **Я сам был своим критично настроенным заказчиком**. Я создавал эту программу для себя, а не для других. Безусловно, я был счастлив, когда мне удалось найти нескольких покупателей, хотя мне и не удалось заработать миллионы, на которые я рассчитывал. Но что бы я ни делал, самым важным было, чтобы продукт работал именно так, как *я* этого хотел.

• **У меня не было плана, только список функциональных возможностей, которыми должен был обладать новый продукт**. Я начал с такой наиболее часто применяемой операции, как ввод транзакций. Затем перешел к менее критичным функциям вроде составления баланса и внесения корректировок. В конце добавил такие необязательные приятные вещи, как подсказки и возможность экспорта данных. Я занимался этим до тех пор, пока мне все не надоело, и я просто не объявил продукт готовым.

• **Процесс создания программы менялся по мере написания кода.** Я начал составлять чек-листы для каждой

процедуры, и эти списки постепенно становились все длиннее. Я никогда до этого не слышал о покомпонентном тестировании, но моя система проверок и двойных проверок была не менее надежной, чем та, которой пользуются пилоты.

Вот, собственно, и все. У меня была мотивация, а также критично настроенный заказчик, при этом отсутствовал предварительный план, но присутствовали дисциплина и самоорганизующийся процесс. Не имело никакого значения, что ранее я никогда ничего подобного не делал. Важно было то, что у меня было страстное желание учиться.

Через десять лет после создания своей бухгалтерской программы я узнал, что часть процесса, который я использовал, теперь вдруг стали называть гибкими методологиями разработки ПО. Еще десятью годами позже я начал писать книгу о компонентах, которых, с моей точки зрения, не хватает гибким методологиям. Как не раз бывало раньше, мне казалось, что этот путь позволит заработать миллионы.

Евангелие от Agile

Вначале инженеры сотворили компьютеры и программное обеспечение. Программное обеспечение же было бесформенно и нефункционально, и тьма царила над пользователями. И инженеры сказали: «Да будет структура». И возникла структура.

И даже в избытке.

За последние пять-шесть десятков лет многие инженеры-компьютерщики озаботились проблемой нестабильности качества при разработке ПО. Она объяснялась тем, что разные команды пользовались разными методами разработки, в основном созданными на коленке. И инженеры окунулись в работу. В результате возник ряд *формализованных* подходов. Родилась профессия **разработчика программного обеспечения**. Отправной гипотезой служило представление, что создание ПО – чисто *инженерная* задача. В результате появилось большое количество моделей, методов, подходов и технических приемов, которые, по идее, должны были помочь программистам повысить качество готовых программ. Но странное дело – при реализации большинства проектов эти методы оказывались неэффективными. Гораздо чаще их результатом становилось возникновение очередной бюрократии. Проекты по разработке ПО занимали столько времени и требовали создания такого коли-

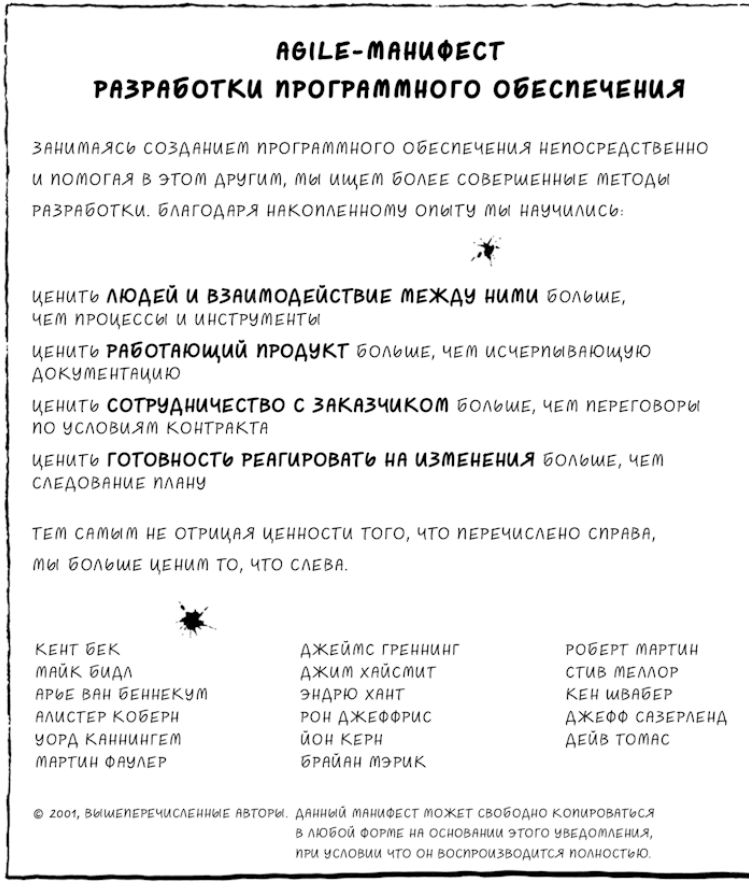
чества документации, что «формальные» требования к продукту успевали по нескольку раз измениться за то время, что проект находился в разработке. Тем временем небольшим командам программистов удавалось выпускать на рынок продукты гораздо более высокого качества, на порядки дешевле и существенно быстрее. На их стороне были энтузиазм, дисциплина, гибкость и методы, адаптируемые под каждый проект. Эволюция произвела на свет динозавров, но вся пища доставалась муравьям.

В начале 1990-х была предложена новая методология под названием **быстрая разработка приложений (Rapid Application Development, RAD)**. В рамках этой методологии наиболее успешным командам разработчиков удавалось сочетать некоторые формализованные методы, позаимствованные у «тяжеловесного» инженерного подхода (контроль за внесением изменений в техдокументацию, инспекции и применение контрольных показателей), с такими продиктованными практикой методами, как создание прототипов, выпуск инкрементных версий ПО и тесное сотрудничество с заказчиком [McConnell 1996]. В результате такого скрещивания формализованных и неформализованных методик возникли первые «легкие» методологии, включая **эволюционное управление разработкой** (**Evo**) (1988), **Scrum** (1995), **методы разработки динамических систем** (**DSDM**) (1995), методы **Crystal** (1997), **Экстремальное программирование (XP)** (1999), **разработка, управ-**

ляемая функциональностью (FDD) (1999), **прагматическое программирование** (1999) и **адаптивная разработка ПО** (2000).

Как следствие внезапного и почти одновременного появления множества методик, статей, книг и семинаров по «легким» методологиям (некоторые даже сравнивали его с Кембрийским взрывом), у лидеров движения возникла идея собраться и обсудить положение дел. В 2001 году они встретились на лыжном курорте в штате Юта. Там и был выбран термин «гибкие методологии» (Agile), заменивший применявшуюся ранее терминологию, а также был создан Agile-манифест разработки ПО (рис. 2.2).

Agile-манифест многими рассматривался в первую очередь как реакция против бюрократического характера существовавших на тот момент формальных подходов, которые были слишком «упорядоченными». Но лишь немногие поняли, что авторы манифеста также выступают против отсутствия дисциплины у программистов, против «хаотических» процессов и низкого качества, которое в то время доминировало на рынке ПО. Лидеры нового движения осознали, что существует средний путь между структурированностью и отсутствием структуры, между упорядоченностью и хаосом. В определенном смысле это была героическая попытка вернуться к более ранней эпохе, когда основными игроками были программисты-первопроходцы, но анархии при этом не было.



AGILE-МАНИФЕСТ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

ЗАНИМАЯСЬ СОЗДАНИЕМ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ НЕПОСРЕДСТВЕННО И ПОМОГАЯ В ЭТОМ ДРУГИМ, МЫ ИЩЕМ БОЛЕЕ СОВЕРШЕННЫЕ МЕТОДЫ РАЗРАБОТКИ. БЛАГОДАРЯ НАКОПЛЕННОМУ ОПЫТУ МЫ НАУЧИЛИСЬ:

ЦЕНИТЬ **ЛЮДЕЙ И ВЗАИМОДЕЙСТВИЕ МЕЖДУ НИМИ** БОЛЬШЕ, ЧЕМ ПРОЦЕССЫ И ИНСТРУМЕНТЫ

ЦЕНИТЬ **РАБОТАЮЩИЙ ПРОДУКТ** БОЛЬШЕ, ЧЕМ ИСЧЕРПЫВАЮЩУЮ ДОКУМЕНТАЦИЮ

ЦЕНИТЬ **СОТРУДНИЧЕСТВО С ЗАКАЗЧИКОМ** БОЛЬШЕ, ЧЕМ ПЕРЕГОВОРЫ ПО УСЛОВИЯМ КОНТРАКТА

ЦЕНИТЬ **ГОТОВНОСТЬ РЕАГИРОВАТЬ НА ИЗМЕНЕНИЯ** БОЛЬШЕ, ЧЕМ СЛЕДОВАНИЕ ПЛАНУ

ТЕМ САМЫМ НЕ ОТРИЦАЯ ЦЕННОСТИ ТОГО, ЧТО ПЕРЕЧИСЛЕНО СПРАВА, МЫ БОЛЬШЕ ЦЕНИМ ТО, ЧТО СЛЕВА.

КЕНТ БЕК
МАЙК БИДЛ
АРЬЕ ВАН БЕННЕКУМ
АЛИСТЕР КОБЕРН
УОРД КАННИНГЕМ
МАРТИН ФАУЛЕР
ДЖЕЙМС ГРЕННИНГ
ДЖИМ ХАЙСМИТ
ЭНДРЮ ХАНТ
РОН ДЖЕФФРИС
ЙОН КЕРН
БРАЙАН МЭРИК
РОБЕРТ МАРТИН
СТИВ МЕЛЛОР
КЕН ШВАБЕР
ДЖЕФФ САЗЕРЛЕНД
ДЕЙВ ТОМАС

Рис. 2.2. Agile-манифест разработки программного обеспечения

Впоследствии группа наиболее авторитетных представи-

телей Agile-движения создала Agile Alliance[4] – некоммерческую организацию, которая ставит себе целью продвижение гибких методологий во всем мире. Возникла целая новая экосистема, состоящая из конференций, консультантов, книг и журналов. В результате процессы разработки программного обеспечения стали Agile с большой буквы А, превратившись в нечто более глубокое, чем просто набор практик, которые можно использовать при разработке софта. Признавая, что проекты по разработке программного обеспечения существуют в области, которая располагается между упорядоченностью и хаосом, Agile-подходы, по сути, превратились в образ жизни.

[4] У этой организации есть свой сайт https://www.agilealliance.org.

Фундаментальные принципы Agile-методологий

В наши дни численность людей, которые разделяют ценности и принципы Agile-методологий, составляет несколько миллионов человек. Опросы подтверждают, что большинство разработчиков программного обеспечения во всем мире придерживаются по крайней мере некоторых из «основных Agile-практик» [VersionOne 2009].

Фундаментальные принципы Agile-методологий были неоднократно описаны, и у многих авторов это получается гораздо лучше, чем у меня. И все же я чувствую необходимость привести в своей книге их краткий обзор. Будучи практиком гибких методологий, я предпочитаю делать все так, как удобно лично мне; поэтому кратко опишу их основные положения, перечислив «семь измерений», в которых «живут» проекты по разработке ПО, – и еще раз вернусь к этой теме в главе 11 «Развитие компетенций».

Люди

Прежде всего Agile-методологии признают за людьми их уникальность и не относятся к ним как к взаимозаменяемым ресурсам. Также признается, что основную ценность пред-

ставляют взаимодействия и сотрудничество между людьми, а не их индивидуальные компетенции. Данный подход также предполагает работу в небольших кросс-функциональных командах, объединяющих людей, выполняющих разные роли (разработчиков, дизайнеров, тестировщиков и так далее). Предпочтительным вариантом будет размещение команды в одном помещении. От команды требуется самоорганизоваться, что означает отсутствие навязываемых извне методов или рабочих процессов. Команде доверяется выполнение определенной работы, исходя из представления, что ее члены знают, как эту работу выполнить, и осознают свою ответственность за результат.

Функциональность

В рамках Agile-методологий признается, что лучшие программные продукты создаются в условиях, когда заказчик максимально вовлечен в процесс разработки. Команда сотрудничает с заказчиком (или его представителем), поддерживая в актуальном состоянии backlog проекта и постоянно обновляя совместные приоритеты. Описание желаемой функциональности осуществляется в предельно кратком виде и детализируется только непосредственно перед началом работы над ней. Простота будет ключом к хорошему дизайну каждой из функциональных возможностей. Полезность данной функциональности оценивается и подтверждается кли-

ентом сразу же после ее создания.

Качество

Качество играет определяющую роль в успехе продукта, поэтому в центре внимания Agile-методологий находится техническое совершенство. Высокий технический уровень обеспечивается посредством разработки через тестирование (написание протокола тестирования готового продукта предшествует созданию собственно программного кода), ревью кода (часто в сочетании с парным программированием), Definition of Done (чек-лист готовности элементов), итеративной разработки (адаптация кода в результате появившихся изменений или других обстоятельств) и рефакторинга (непрерывная оптимизация кода даже при отсутствии изменений в функциональности). Сторонники гибких методологий признают необходимость последовательного улучшения дизайна; под этим понимается, что в начале проекта архитектура продукта не разрабатывается в деталях (а только в самом базовом виде) и выявляется при дальнейшем развитии проекта.

Инструменты

Сторонники Agile-методологий считают, что инструмен-

ты – наименее важный из факторов, влияющих на успешность проекта. И тем не менее инструменты часто упоминаются и продвигаются в литературе по гибким методологиям. Опытные Agile-команды предпочитают инструменты, позволяющие осуществлять ежедневные сборки, непрерывную интеграцию и автоматическое тестирование. Команды нуждаются в мотивации, поэтому повторяющиеся скучные операции должны быть автоматизированы. Многие сторонники гибких методологий в качестве обязательного условия выдвигают наличие поддерживающей «среды обитания» в виде открытого офисного пространства, а также информационные радиаторы (к ним относятся, например, доска задач и диаграммы сгорания задач). Предназначение инструментов в контексте Agile-методологий – усиливать мотивацию, коммуникации и сотрудничество внутри команды.

Время

У Agile-методологий особые отношения со временем. В Agile-проектах даты поставки, бюджеты и крайние сроки могут устанавливаться почти произвольно. Программное обеспечение создается короткими отрезками, часто в рамках тайм-боксов или «спринтов», и поставляется в виде инкрементных релизов; при этом каждый из релизов сам потенциально является готовым к поставке продуктом. Это позволяет владельцам бизнеса управлять графиком проекта, сдвигая

дату сдачи готового продукта вперед или назад в зависимости от того, в какие сроки они хотят сделать доступной ту или иную функциональность. Тем временем команда стремится найти для себя оптимальную скорость разработки, которую она сможет поддерживать практически бесконечно.

Ценность

Одной из важнейших причин создания Agile-манифеста было стремление подчеркнуть важность своевременной реакции на изменения. Среда, в которой функционирует программный продукт, никогда не бывает статичной. Функциональность, которая еще вчера представляла собой значительную ценность, завтра может оказаться бесполезной, *включая* функциональность, которая уже имеется в версиях продукта, переданных заказчику. Разработчики, практикующие гибкие методологии, стараются справиться с этой проблемой, предпочитая короткие циклы разработки и обратной связи. Смысл частых релизов программного продукта не только в том, чтобы получить обратную связь от пользователей и учесть ее в последующем процессе разработки, но и в том, чтобы предоставить пользователям новую функциональность как можно скорее после выявления их потребности в ней, тем самым повышая ценность ПО для клиента.

Процесс

Несмотря на то, что доминирующая парадигма Agile-методологий – люди, а не процессы, это совсем не означает, что последние не важны. Наиболее важными процессами в этом контексте будут минимальное планирование (или планирование методом набегающей волны), ежедневное личное общение (часто это происходит в формате ежедневных стендапов) и мониторинг хода проекта через оценку работающего продукта (по функциональным возможностям, принятым клиентом). Приверженцы гибких методологий также признают необходимость непрерывного улучшения, в ходе которого процессы разработки подвергаются регулярной переоценке и перенастройке посредством анализа и ретроспектив (ретроспективных совещаний).

Конфликт

Все вышеизложенное отражает мое понимание сущности Agile-методологий. Естественно, это не более чем моя точка зрения. Некоторые разработчики, практикующие гибкие методологии, могут не согласиться с приведенным мной кратким описанием. Но это вытекает из самого характера Agile-методологий. Я даже мог бы включить понятие «конфликт»

в качестве восьмого измерения, присущего этим методологиям. Как вы увидите позже, наличие внутреннего конфликта – естественное свойство сложных систем и необходимое условие для креативности и инноваций. Великая честь оказаться среди людей, которым ничто не доставляет такого удовольствия, как возможность совершенствовать друг друга.

Методологии, конкурирующие с Agile

Редко когда попадаются игры, не предусматривающие конкуренцию между игроками, и лишь немногие системы обходятся без конфликта. Без расхождения во взглядах между разными людьми жить было бы не так интересно. К счастью, в мире гибких технологий предостаточно здоровой конкуренции, например между Scrum и Экстремальным программированием, Scrum и канбаном и даже между Scrum и Scrum![5] Но гибкие методы разработки не будут здесь единственными игроками. Существует несколько сильных и многообещающих конкурирующих подходов, в основе которых лежат идеи, аналогичные идеям Agile-методологий, дополняющие их, а часто и входящие с ними в прямое противоречие.

Одними из самых крупных конкурентов будут методологии **бережливой разработки программного обеспечения (Lean software development)**, переносящие идеи бережливого производства в область разработки ПО. Семь принципов бережливого производства [Poppendieck 2009: 193] основываются на четырнадцати принципах Дао Toyota (философии управления компании Toyota) и четырнадцати принципах менеджмента Э. Деминга. Между Agile- и Lean-методологиями много общего, поэтому часто они играют на

[5] Сайт http://www.scrum.org был создан основателем Scrum Кеном Швабером.

одной стороне, ими занимаются одни и те же эксперты, у них одни и те же фанаты, а их развитие освещается в одних и тех же блогах, журналах и ТВ-шоу. С управленческой точки зрения Lean-методологии – с их акцентом на сокращении непродуктивных затрат и оптимизации систем в целом – внесли большой вклад в развитие гибких методологий. Хотя бережливые методологии разработки ПО возникли на несколько лет позднее Agile, они сравнялись с ними по числу консультантов, коучей, профессиональных консорциумов[6] и проводимых конференций.

Менее крупным, но весьма способным игроком будет также **движение за мастерство программирования**

[6] На данный момент The Lean Software & Systems Consortium реорганизован в The Lean Systems Society, сайт организации http://leansystemssociety.org. – *Прим. ред*.

Конец ознакомительного фрагмента.