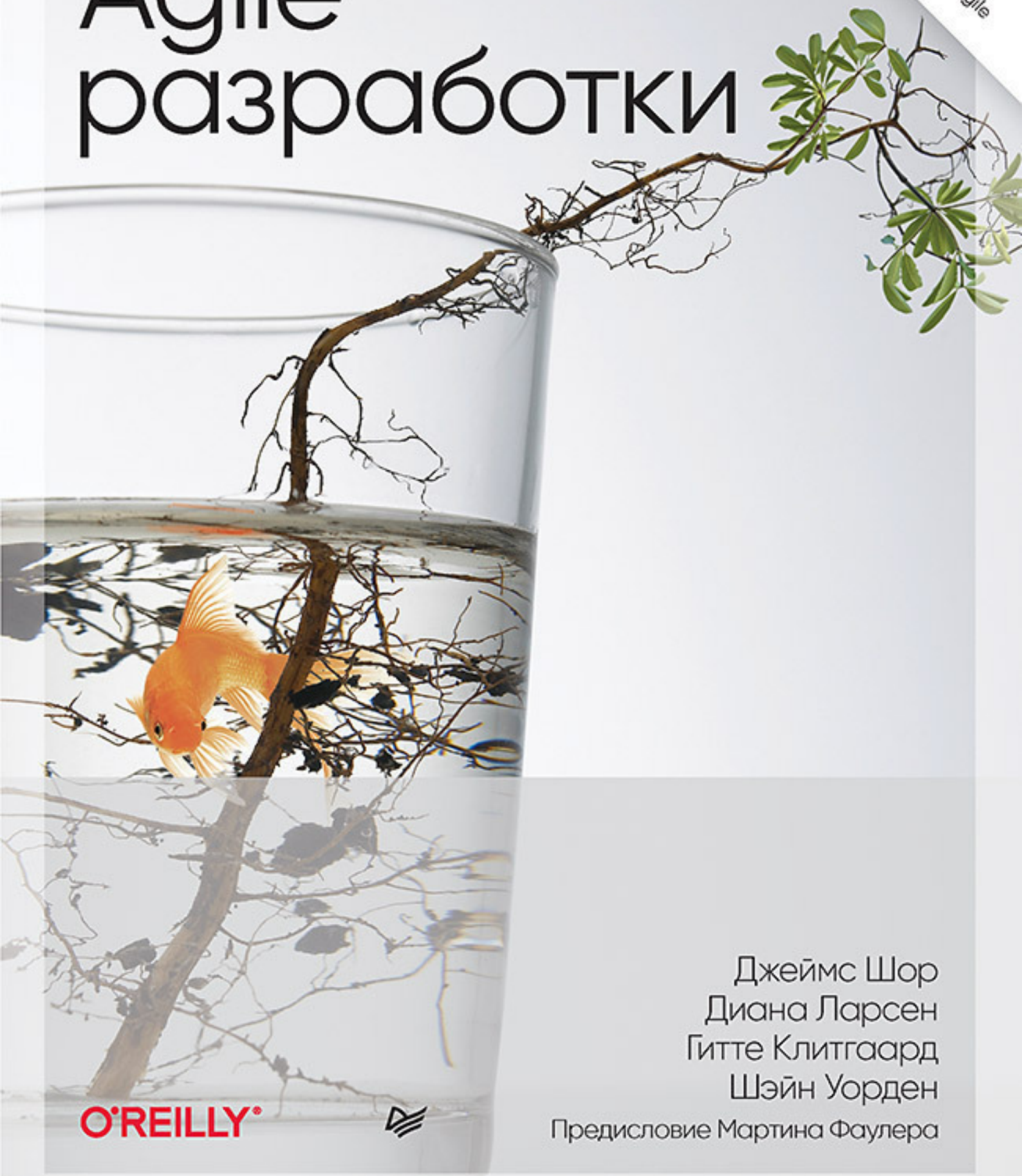
2-е издание
На 100 % соответствует современным стандартам Agile

Искусство

Agile разработки

Теория и практика гибкой разработки ПО



Джеймс Шор
Диана Ларсен
Гитте Клитгаард
Шэйн Уорден

Предисловие Мартина Фаулера

O'REILLY®

Бестселлеры O’Reilly (Питер)

Шэйн Уорден

Искусство Agile-разработки. Теория и практика гибкой разработки ПО

«Питер»
2022

УДК 004.4:005.8
ББК 32.973-018

Уорден Ш.

Искусство Agile-разработки. Теория и практика гибкой разработки ПО / Ш. Уорден — «Питер», 2022 — (Бестселлеры O'Reilly (Питер))

ISBN 978-5-4461-2386-5

Большинство компаний, разрабатывающих ПО, якобы используют Agile, но на самом деле не понимают, что это такое Agile. Хотите повысить гибкость своей команды? В книге вы найдете четкие, конкретные и подробные рекомендации о том, что, как и почему следует делать, а когда стоит пойти на компромиссы.

Джеймс Шор предлагает реальные решения по освоению, планированию, разработке и управлению, основанные на более чем двадцатилетнем опыте Agile. Он объединяет актуальные идеи экстремального программирования, Scrum, Lean, DevOps и многих других в единое целое. Узнайте, как успешно внедрить гибкую разработку в вашей команде и организации, или разберитесь, почему Agile вам не подходит. В формате PDF A4 сохранен издательский макет книги.

УДК 004.4:005.8
ББК 32.973-018

ISBN 978-5-4461-2386-5

Содержание

Джеймс Шор, Шэйн Уорден
Искусство Agile-разработки
Теория и практика гибкой разработки ПО

Отзывы

Автор «Искусства Agile-разработки» смог совершить, казалось бы, невозможное, собрав все материалы по обширнейшей теме разработки современного программного обеспечения в понятную и увлекательную книгу. Новички, лишь начинающие изучать итеративные процессы, найдут в ней отличный обзор популярных практик. Людям, заблудившимся в дебрях мудреных процессов «масштабизации Agile», она предлагает хорошие идеи выхода из этого ада. Первое издание оказало огромное влияние на мою карьеру два десятилетия назад, и, уверен, второе издание аналогичным образом поможет миллионам разработчиков улучшить свои методы поставки ПО.

Гойко Аджич, автор книг Running Serverless, Impact Mapping и Specification by Example

В этой книге есть все: от кода до поставки готового продукта. Заработанные десятилетиями тяжелого труда знания превратились в легкочитаемый и доступный для восприятия материал – обязательный к прочтению для всех, кто работает с командой разработчиков программного обеспечения или в ней.

Ави Кесснер, инженер, Forter

Эта книга получит свое постоянное место на моей книжной полке.

Кришна Кумар, основатель и CEO, Exathink Research

Первое издание этой книги заворожило меня до такой степени, что до сих пор стоит на моей книжной полке в качестве справочника. Второе издание сохраняет шарм своего предшественника и содержит еще больше знаний, накопленных за последнее десятилетие.

Бенджамин Мускалла, старший инженер-программист, GitHub

Одна из самых полных книг по Agile-разработке программного обеспечения, которые я когда-либо читала. Очень прагматичная, с яркими примерами, легко применимыми к любому проекту разработки ПО независимо от технологического стека, размера команды или предметной области отрасли. Определенно жемчужина, стоящая того, чтобы держать ее под рукой.

Луиза Нуньес, менеджер программ, Thoughtworks

Это моя любимая книга об Agile. Она охватывает технические и управленческие темы. Я использую ее на своих занятиях и всегда рекомендую своим клиентам.

Николас Паес, инженер-программист и преподаватель, Университет Буэнос-Айреса

Джеймс подробнейшим образом описывает собственный опыт в Agile-разработке ПО. Простыми словами, связывая теорию с практикой.

Кен Пью, главный консультант, автор книги Prefactoring: Extreme Abstraction, Extreme Separation, Extreme Readability

Тысячи книг об Agile. Какую из них прочитать? Я советую вам эту. Джеймс работает с Agile с первых дней его появления и знает свое дело. Книга разбирает весь бессмысленный мусор в нашей отрасли, окружающий это понятие, и предлагает тщательно выверенный, целостный подход. Этот подход не будет быстрым и легким, но он того стоит. Мне понравилась книга «Искусство Agile-разработки». Это книга с характером!

Бас Водде, соавтор LeSS

Джеймс Шор полностью обновил «Искусство Agile-разработки», дополнив книгу новыми инструментами, методами и уроками прошедшего десятилетия. Эта жемчужина среди книг поможет вам превратить ваш стиль работы в действительно эффективный Agile-подход.

Билл Уэйк, XP123, LLC

Моей семье

Предисловие

Когда мы писали Манифест Agile для разработки программного обеспечения, нашими сторонниками было незначительное меньшинство людей, пытавшихся изменить отрасль. Сейчас, 20 лет спустя, «agile» – это мейнстрим. Я пишу «agile» в кавычках намеренно: множество людей говорят, что они занимаются разработкой программного обеспечения по Agile, в самом деле искренне в это веря, однако их действия мало напоминают то ви́дение, которым мы поделились со всеми два десятилетия назад.

Правда в том, что для работы по Agile требуется целая сеть взаимосвязанных практик, охватывающая как управление, так и техническую работу по созданию ПО. Многие из этих практик, особенно технические, далеко не всем хорошо понятны или не изучаются широко. Следовательно, слишком многие люди имеют искаженное представление о том, что могло бы быть очень эффективным способом создания программных продуктов.

Джеймс Шор был одним из пионеров, вставших на тропу экстремального программирования (Extreme Programming, XP), центрального элемента движения Agile. Первое издание этой книги было отличным руководством для команд, показывавшим им, что нужно знать для правильного выполнения Agile-процессов. Позже Джеймс вместе с Дианой Ларсен продолжили работу над созданием модели Agile Fluency™, которая объединила в себе их опыт развития навыков в использовании подходов Agile. Модель показывает, что простое применение техник проектного управления, часто именуемых базовым подходом Scrum, дает некую ценность, фокусируя внимание на потребностях клиентов, но не учитывает технических навыков, необходимых для того, чтобы полностью раскрыть преимущества высокой производительности и надежности, которых добиваются многие команды.

Эта позиция непосредственно определяет структуру книги, значительная часть которой посвящена тому, как сфокусироваться на создании реальной ценности и как надежно поставить ее потребителю продукта. «Сфокусироваться на ценности» означает понимать потенциал командной работы, развивать навыки адаптивного планирования и тесно сотрудничать с заказчиками и конечными пользователями готового ПО. В понятие «надежно поставить ценность» входят основные технические практики тестирования, рефакторинга, дизайна и коллективной разработки. Оно также учитывает парадоксальное наблюдение, что создание программного обеспечения, обладающего высоким внутренним качеством, снижает затраты и увеличивает скорость поставки продукта. Сочетание культуры DevOps и непрерывной поставки (continuous delivery) дает возможность быстро вводить в эксплуатацию множество функций, а это, в свою очередь, позволяет командам понять, что является ценным в программном обеспечении, наблюдая, как оно используется на практике.

Двадцать лет назад мне повезло найти свое место в компании Thoughtworks, где наши команды используют такие навыки, чтобы создавать для наших клиентов новые программные продукты и заменять ими устаревшие. Как и Джеймс, мы обнаружили, что техники экстремального программирования служили надежной основой для нашей работы, и мы весьма успешно применяли их в течение последних двух десятилетий. Поэтому я очень рад, что Джеймс переписал свою книгу, добавив в нее опыт еще одного десятилетия своей работы в роли коуча. В результате получился прочный фундамент для изучения тех навыков, которые нам так помогли. Как и все действительно стоящее, обучение займет время, и на вашем пути будут разочарования. Но этот путеводитель может вам помочь – избавив от безрезультативных церемоний и приведя к источнику силы, которую мы с Джеймсом почувствовали, впервые применив эти методы много лет назад.

Мартин Фаулер, Chief Scientist, Thoughtworks

Введение

Вопрос: Что нужно делать, чтобы выступить в Карнеги-холле?

Ответ: Практиковаться, практиковаться и еще раз практиковаться.

Я хочу помочь вам овладеть мастерством Agile-разработки.

Agile, как и любой командный подход к разработке программного обеспечения, – в своей основе *человеческое* искусство, зависящее от капризов людей и их взаимодействия. Чтобы мастерски овладеть Agile-разработкой, вам нужно научиться ежеминутно оценивать множество возможностей и интуитивно выбирать лучшее направление действий.

Как можно научиться такому сложному искусству? Практикуясь!

Во-первых, эта книга – руководство. Она содержит подробное описание одного из способов практического применения Agile. Книга основана на экстремальном программировании, но также берет идеи и практики из Scrum, Kanban, DevOps, Lean Software Development, Lean Startup и др. В конечном счете это практическое пособие, которое позволит вам успешно внедрить Agile-разработку в вашу команду и организацию или поможет обнаружить, что Agile не подходит в вашей ситуации.

Во-вторых, эта книга призвана помочь вам овладеть *мастерством* Agile-разработки. Освоить Agile означает выйти за рамки книги рецептов. Разработка программного обеспечения – слишком чувствительный к контексту процесс, чтобы какой-то один подход подошел вам полностью, и слишком богатый нюансами, чтобы какая-то книга могла научить вас, как его освоить. Мастерство приходит изнутри – благодаря опыту и интуитивному пониманию природы волн, вызванных брошенным в воду камнем вашего выбора.

Я не могу спрогнозировать, какую волну поднимет ваш выбор в вашей организации. Я и не пытаюсь. Вы должны сами определить нюансы и достичь понимания. Это единственный способ овладеть искусством. Следуйте практикам. Смотрите, что происходит. Думайте, почему они сработали… или не сработали. Потом повторяйте. Что было таким же? Что получилось по-другому? Почему? Потом делайте это снова. И снова.

Поначалу вам может быть трудно понять, как использовать каждую из практик. Они могут выглядеть простыми на бумаге, но сложными в реальном применении. Продолжайте практиковаться, пока не станет легче.

По мере того как применять Agile станет проще, вы обнаружите, что некоторые мои советы для вас не работают. Поначалу вы не сможете различить, в чем проблема: в инструкциях, которые я дал, или в том, как вы их выполняете. Продолжайте практиковаться, пока не почувствуете уверенность. Как только это произойдет, нарушьте правила. Измените мои инструкции так, чтобы они работали лучше в вашей конкретной ситуации. В каждой практике есть подраздел «Альтернативы и эксперименты», содержащий идеи для исследований.

Однажды правила перестанут вас интересовать. В конце концов, Agile не о том, чтобы следовать правилам. Тогда вы подумаете: «Речь о простоте и обратной связи, общении и доверии. О поставке ценности и смелости делать правильные вещи в нужное время». Вы станете мгновенно оценивать множество возможностей и интуитивно выбирать лучшее направление действий.

Когда этот момент настанет, отдайте книгу (пусть и с загнутыми уголками страниц и, возможно, слегка потрепанную) кому-нибудь еще, чтобы эти люди тоже смогли овладеть искусством разработки Agile.

Для прагматиков

А что, если вы не хотите овладевать так называемым «искусством»? Что, если вы хотите просто разрабатывать хорошее ПО?

Не волнуйтесь. Эта книга и для вас тоже. На основе моего многолетнего опыта Agile-разработки я создал единый, четко определенный, комплексный подход.

Это позволяет мне использовать простой язык. Я даю много практических советов. Я честно пишу, когда мой подход не будет работать и какие альтернативы можно рассмотреть в таком случае. Глава 2 поможет вам начать.

Есть и обратная сторона обсуждения только одного подхода: ни один подход не применим ко всем случаям. Мой совет может быть непригоден для вашей команды или организации. Прочтите главы 4 и 5, чтобы понять, на каких общих условиях возможен успех, и изучите подраздел «Предварительные требования» каждой практики.

Не стоит сразу думать, что какая-либо конкретная практика вам не подходит. Некоторые практики, описанные в этой книге, противоречат здравому смыслу или просто не кажутся интересными. Большинство из них лучше всего работают в сочетании с другими. Если можете, то попробуйте применять практики, как написано, в течение нескольких месяцев, получите реальный опыт их работы в ваших условиях и *потом* измените их.

Я применяю эти идеи в реальной работе уже более 20 лет. В правильной среде они действительно работают. Agile оказывается более увлекательным и успешным, чем любой другой подход к разработке программного обеспечения, который я пробовал. Присоединяйтесь.

Что нового во втором издании

Первое издание было полностью переработано. Во втором издании сохранен приземленный, практический подход, свойственный первому изданию, как и большинство приводившихся в нем практик. Но почти все они были переписаны с учетом 14 лет прогресса Agile, не говоря уже о моем собственном 14-летнем опыте.

Я полностью переработал книгу, чтобы она позволяла обеспечить постепенное внедрение Agile и лучше отражала реальное использование командами. Принципы и настройки, обсуждавшиеся в части III первого издания, были распределены между практиками, чтобы сделать их более заметными и понятными, и я внес в каждую практику предложения по экспериментам.

Среди наиболее заметных дополнений:

• подробное руководство по внедрению Agile и адаптации этого внедрения к потребностям вашей компании, основанное на модели Agile Fluency™[1], которую я создал вместе с Дианой Ларсен;

• новая глава о масштабировании Agile, основанная на моем опыте помощи большим и маленьким компаниям;

• новая глава о DevOps, содержащая новый материал о работе в сфере эксплуатации и безопасности, а также вдохновленные темой DevOps обновления по остальной части книги;

• руководство по внедрению Agile для работы с удаленными командами; много новых практик, историй и идей; а также слишком много других улучшений и изменений, чтобы упомянуть их все.

[1] Agile Fluency – зарегистрированный товарный знак Agile Fluency Project LLC.

Для кого предназначена книга

Эта книга для каждого, кто уже работает с командой Agile или надеется работать в будущем. Сюда входят, конечно же, программисты, а также менеджеры, руководители высшего звена, эксперты в предметных областях, тестировщики, продакт-менеджеры, руководители проектов, архитекторы, сотрудники отделов эксплуатации и отделов безопасности, дизайнеры и бизнес-аналитики. Команды Agile кросс-функциональны; книга отражает этот факт.

Книга представляет собой краткий справочник, но ее также можно читать и подряд, от корки до корки. Каждая практика в частях II–IV предназначена для самостоятельного чтения. Блоки «См. также» и ссылки на источники помогут вам сопоставить информацию. Вдобавок печатное издание можно открыть в любом месте и быстро просмотреть интересующую информацию. Можете пролистать книгу и остановиться, чтобы прочитать какую-то врезку более внимательно, если она привлечет ваше внимание.

Если вы менеджер или руководитель высшего звена, который хочет понять, как Agile может или должен работать в вашей компании, то прочитайте часть I. Если вы руководитель командного уровня, то прочтите еще и раздел «Менеджмент» главы 10 и, возможно, другие практики, рассматриваемые в той же главе.

Если вы член команды или менеджер, заинтересованный в том, чтобы внедрить Agile в свою компанию или улучшить уже работающую в ней практику Agile, то прочитайте всю книгу целиком. Часть I поможет вам понять, как представить идеи Agile коллегам. Остальной материал книги поможет разобраться, как воплотить Agile в жизнь.

Если вы часть Agile-команды и просто хотите получить знания, которые позволят выполнять вашу работу, то можете сосредоточиться на практиках, изложенных в частях II и III. Начните с главы 1, чтобы получить общую картину, затем перейдите к тем практикам, которые использует ваша команда.

Если вы не являетесь частью Agile-команды, но работаете с одной из них, то посоветуйтесь с ними, что почитать. Продакт-менеджеры, владельцы продукта, дизайнеры могут начать с главы 8 и раздела «Цель» главы 7. Сотрудникам отделов безопасности и отделов эксплуатации рекомендую разделы «Сборка для эксплуатации» главы 15, «Обнаружение слепых зон» и «Анализ инцидентов» главы 16. Тестировщики, ознакомьтесь с главой 16.

Если вам просто любопытно узнать об Agile, то начните с главы 1. После этого ознакомьтесь с частями II–IV. Начните с наиболее интересных для вас практик. Можете читать их в любом порядке.

О приглашенных авторах

Эта книга создана в соавторстве с несколькими выдающимися людьми. Гитте Клитгаард написала раздел «Безопасность» главы 7, профессионально осветив тему психологической безопасности. Диана Ларсен написала разделы «Динамики команды» и «Устранение препятствий» главы 11, поделившись своим многолетним опытом организационного и командного развития. Шейн Уорден, мой соавтор первого издания, не смог написать новый материал для второго, но остался надежным и ценным советчиком, и наша работа над первым изданием легла в основу этой книги.

Гитте Клитгаард

Гитте Клитгаард – Agile-коуч, инструктор и наставник, специализирующаяся на помощи организациям в формировании психологической безопасности и ответственности. Гитте сразу переходит к сути дела и помогает людям стать самими собой и таким образом достичь успеха.

Ее вклад в сообщество включает организацию встреч для тренеров и выступления на конференциях, где она поднимает темы психического здоровья и психологической безопасности и делает их доступными для общего понимания. Гитте создает безопасную и уважительную атмосферу на работе и за ее пределами. Она умеет слушать и вовлекать в обсуждение более тихие голоса и малые сообщества.

В свободное время Гитте собирает LEGO, коллекционирует фигурки Йоды и поддерживает связь с друзьями со всего земного шара, многих из которых она считает своей второй семьей.

Гитте является владельцем Native Wired и руководила преобразованиями в таких компаниях, как LEGO, Spotify и Mentimeter.

Диана Ларсен

Более 20 лет Диана Ларсен вносила свой вклад в формирование основ и расширение идей Agile, а также в практику создания и развития квалифицированных команд. Диана – соавтор книг *Agile Retrospectives*[2], *Liftoff, 2nd ed., Five Rules for Accelerated Learning* и электронной книги *The Agile Fluency Model: A Brief Guide to Success with Agile*; кроме того, сейчас пишет две новые книги. Будучи главным коучем, консультантом, координатором, спикером и наставником, она продолжает вносить свой вклад и соответствует своему званию в проекте Agile Fluency – «Главное связующее звено» (Chief Connector). Живет в Портленде, на верхнем левом побережье США.

Шейн Уорден

Шейн Уорден – руководитель инженерно-технической службы и писатель, в частности, соавтор первого издания этой книги, а также книги *Masterminds of Programming*[3]. В свободное от работы время он помогает находить животным хорошие дома.

[2] *Дерби Э., Ларсен Д.* Agile ретроспектива. Как превратить хорошую команду в великую.

[3] *Бьянкуцци Ф., Уорден Ш.* Пионеры программирования: диалоги с создателями наиболее популярных языков программирования.

Условные обозначения

В книге используются следующие условные обозначения.

Аудитория
В этих блоках указывается целевая аудитория каждой практики Agile.

См. также
В этих блоках приводятся связанные практики.

Курсив
Курсивом выделены новые термины и важные понятия.

Шрифт одной ширины

Используется для текста программ, а также внутри абзацев для обозначения таких элементов, как переменные и функции, базы данных, типы данных, переменные среды, операторы и ключевые слова, имена файлов и их расширений.

Жирный шрифт одной ширины
Показывает добавленный код в работающих примерах кода.

Перечеркнутый шрифт одной ширины
Показывает удаленный код в работающих примерах кода.

Шрифт без засечек

Используется для обозначения URL, адресов электронной почты, названий кнопок, каталогов.

Использование примеров кода

Дополнительные материалы доступны для скачивания по ссылке **https://www.jamesshore.com/v2/books/aoad2**.

Пожалуйста, напишите по адресу электронной почты **bookquestions@oreilly.com**, если у вас технический вопрос или проблема с использованием материалов.

Эта книга призвана помочь в решении ваших задач. В общем случае все примеры кода из книги вы можете использовать в своих программах и в документации. Вам не нужно обращаться в издательство за разрешением, если вы не собираетесь воспроизводить существенные части кода, или если вы разрабатываете программу и используете в ней несколько фрагментов кода из книги, или если будете цитировать эту книгу либо примеры из нее, отвечая на вопросы. Вам потребуется разрешение от издательства O'Reilly, если вы хотите продавать или распространять примеры из книги либо хотите включить существенные объемы программного кода из книги в документацию вашего продукта.

Мы рекомендуем, но не требуем добавлять ссылку на первоисточник при цитировании. Под ссылкой на первоисточник мы подразумеваем указание авторов, издательства и ISBN.

Получить разрешение на использование значительных объемов программного кода примеров из этой книги можно по адресу **permissions@oreilly.com**.

Благодарности

Эта книга вдохновлена бессчетным количеством источников. На протяжении всей книги я поблагодарил скольких мог, но наверняка забыл кого-то. (Пожалуйста, примите мои извинения.) Я хочу выразить особую признательность Кенту Беку, Рону Джеффрису и Уорду Каннингему за то, что они создали экстремальное программирование, с которого я начал мое Agile-путешествие. Алистер Коберн и его круглый стол по программному обеспечению, как и бурные споры и обсуждения Вики-страниц C2 Уорда Каннингема, помогли определить направление в начале этого пути. Благодарю также Диану Ларсен, с которой я работаю много лет – ее точка зрения так хорошо уравновешивает мою. И конечно, Мартина Фаулера, чей вдумчивый стиль письма много лет вдохновляет меня.

Поддержка этого издания со стороны O'Reilly была просто исключительной. Я выражаю огромную благодарность Гэри О'Брайену, моему редактору, обеспечивавшему постоянную обратную связь и поддержку. Благодарю также Мелиссу Даффилд, которая помогла сделать так, чтобы книга пользовалась успехом; Райана Шоу, убедившего меня в том, что пришло время для второго издания; Дебору Бейкер, подготовившую издание ранних выпусков книги; Сюзанну Хьюстон, которая позаботилась о том, чтобы люди узнали о книге; Ника Адамса и команду Tools издательства O'Reilly, которые выстроили производственный процесс и успешно разобрались с моими запутанными и придирчивыми требованиями к оформлению; Кристофера Фоучера, который руководил превращением «сырой рукописи» в «законченную книгу»; Тонью Трибулу и Стефани Инглиш, исправивших мои грамматические чудачества; Кейт Дулли, превратившую мои наброски от руки в понятные рисунки.

Что касается обратной связи, особую признательность выражаю моим рецензентам. Рецензирование было открытым, и мы получили более 700 электронных писем от десятков людей. Практически каждое было содержательным и полезным и в результате улучшило книгу. Кроме того, благодарю тех людей, которые ответили на мои прямые запросы на рецензию. Спасибо всем: Адриану Саттону, Энтони Уильямсу, Ави Кесснер, Басу Водде, Бенджамину Мускалла, Биллу Уэйку, Брэду Эпплтону, Си Кейту Рэйю, Си Джею Джеймсону, Кристиану Дювану, Дэйвиду Пулу, Диане Ларсен, Диего Фонтдевиле, Эмили Бач, Эрику Петерсону, «Эвану М.», Францу Амадору, Джорджу Динвидди, Гойко Аджичу, Джейсону Йипу, Джеффу Григгу, Джеффу Паттону, Джеффри Палермо, Йохану Алуддену, Кену Пью, Кришне Кумару, Лиз Кеог, Луизе Нуньес, Марсело Лопесу, Маркусу Гертнеру, Мартину Фаулеру, Михалу Свободе, Николасу Паесу, Полу Стивенсону, Петеру Гравесу, Реувену Ягелю, Рикардо Майерхоферу, Рону Джеффрису, Рону Квартелу, Саре Хоран Ван Треесе, Стиву Бементу, Томасу Джей Оуэнсу, Тодду Литтлу и Уорду Каннингему.

Особая благодарность тем рецензентам, которые пошли еще дальше, прочитав и прокомментировав почти каждую из частей книги: Басу Водде, Биллу Уэйку, Кену Пью, Мартину Фаулеру и Томасу Джей Оуэнсу.

Процесс открытого рецензирования пошел на пользу и первому изданию, и все его преимущества, в свою очередь, отразились и на этой книге. Спасибо Адриану Ховарду, Адриану Саттону, Энн Баркомб, Энди Лестеру, Энтони Уильямсу, Басу Водде, Биллу Капуто, Бобу Коррику, Брэду Эпплтону, Крису Уилеру, Кларку Чингу, Дади Ингольфссону, Диане Ларсен, Эрику Петерсену, Джорджу Динвидди, Илье Прейсу, Джейсону Йипу, Джеффу Олферту, Джеффри Палермо, Джонатану Кларке, Кейт Рэй, Кевину Рутерфорду, Ким Грэсман, Лизе Криспин, Марку Уайтэ, Николасу Эвансу, Филиппе Антрасу, Рэнди Коулмэн, Роберту Шмитту, Рону Джеффрису, Шейну Доуну, Тиму Хоутону и Тони Бирну за их множественные комментарии и Брайану Марику, Кену Пью и Марку Стрейбеку за их замечания по готовому черновику.

И наконец, еще раз спасибо моей жене Ниру. В этот раз ты знала, на что шла, и все же без колебаний поддерживала меня. Без тебя я бы не смог это сделать.

От издательства

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

Мы будем рады узнать ваше мнение!

На сайте издательства **www.piter.com** вы найдете подробную информацию о наших книгах.

I
Улучшая гибкость

Глава 1. Что есть Agile

Agile везде. И, как ни парадоксально, нигде.

Двадцать лет назад локомотив Agile с ревом ворвался в сознание разработчиков программного обеспечения. За это время количество компаний, называющих себя Agile, возросло на порядки. А сколько команд *действительно* применяет Agile-подход в своей работе? Не так много. Agile, легко повторяемое название, пользуется огромным успехом. А как насчет идей, стоящих за названием? Вообще-то большинство из *них* игнорируется.

Исправим это.

Происхождение Agile

В 1990-е считалось, что разработка ПО находится в кризисе. Это явление так и называлось: «Кризис программного обеспечения». Проекты разработки выходили за рамки бюджетов, задерживались, не отвечали требованиям, и (согласно часто цитируемому и зловеще именуемому «Отчету о Хаосе» (CHAOS Report)) почти треть их была полностью отменена [Standish1994].

Agile не был ответом на этот кризис. Отнюдь нет. Agile стал *ответным подходом*.

Чтобы взять под контроль разработку ПО, большие организации придумали высокодетализированный процесс, точно определявший, как ПО должно создаваться. Все жестко контролировалось, чтобы исключить возможность ошибки. (Во всяком случае, в теории.)

Сначала бизнес-аналитики должны были опрашивать стейкхолдеров (stakeholders, заинтересованные стороны) и документировать системные требования. Далее архитекторы программного обеспечения должны были изучить документы с требованиями и создать подробную проектную документацию, определяющую каждый компонент системы и их взаимосвязь друг с другом. Затем программисты должны были конвертировать проектную документацию в код. В некоторых организациях такое выполнение механического перевода считалось низкоквалифицированной работой.

Тем временем руководители тестирования должны были создавать планы тестирования, используя те же документы, и когда кодирование завершалось, бесчисленные специалисты по обеспечению качества должны были вручную выполнять эти планы, фиксируя отклонения как дефекты. По завершении каждой фазы все должно было быть задокументировано, проверено и подписано.

Подход, основанный на фазах, получил название водопадной (waterfall development) или каскадной разработки (phase-gate development)[4]. Если вам это кажется похожим на нелепое пугало – считайте, что вам повезло. Не все команды применяли в 1990-х перегруженный документацией каскадный процесс, но он широко признавался как логичный и разумный метод работы. Конечно, нужно было определить требования, разработать проект, затем выполнить реализацию и следом – тестирование. Конечно, нужно было документировать каждую фазу. Это была *дисциплина*. Это была *инженерия*. Как еще можно было добиться успеха?

---

[4] Источником появления подхода Waterfall часто ошибочно считают статью Уинстона Ройса, написанную в 1970 году. Однако применение фазового подхода восходит еще к 1950-м, а статья Ройса не пользовалась широкой известностью до конца 1980-х, когда к ней стали прибегать для описания того, что люди уже давно делали [Bossavit2013, chapt. 7].

Рожденный из кризиса

Крупные компании расписывали свои процессы в мельчайших деталях. Роли, ответственности, шаблоны документов, языки моделирования, советы по контролю за изменениями… каждый аспект разработки строго регламентировался и контролировался. Если проект заканчивался провалом (а согласно CHAOS Report, менее одной шестой доли проектов были успешны), причиной считалась недостаточная детализация процесса: нужно больше документов, больше согласований. Это порождало огромное количество документации. Мартин Фаулер описывал это в своей статье *The Almighty Thud* [Fowler1997].

Этот стиль работы был далеко не лучшим. Он был бюрократическим и бесчеловечным. Казалось, знания и навыки не так важны, как соблюдение процессов. Программисты чувствовали себя легко заменимыми винтиками бездушной машины. И даже она работала не очень хорошо.

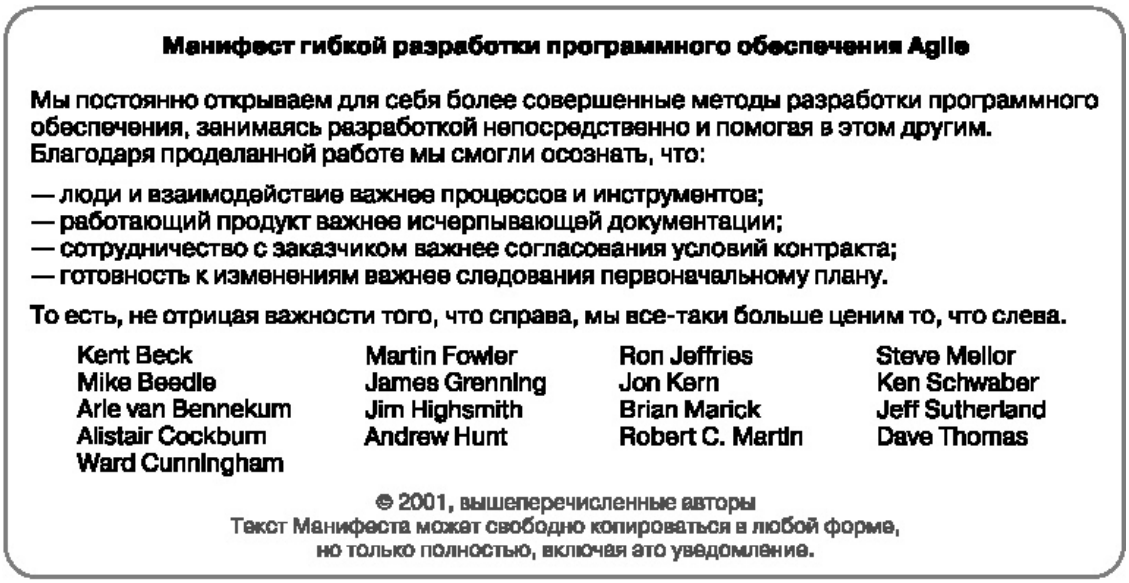
И тогда несколько человек создали более простые, изящные и менее директивные методы разработки программного обеспечения. Они назывались легковесными в отличие от тяжеловесных, использовавшихся большими компаниями. Эти методы носили такие названия, как адаптивная разработка программного обеспечения (Adaptive Software Development), Crystal, разработка на основе функциональности (Feature-Driven Development, FDD), метод разработки динамических систем (Dynamic Systems Development Method, DSDM), экстремальное программирование (XP) и Scrum.

К концу 1990-х эти методы стали привлекать серьезное внимание. Экстремальное программирование, в частности, вызвало вспышку массового интереса среди программистов. В 2001 году 17 сторонников легковесной методологии встретились на горнолыжном курорте в штате Юта, чтобы обсудить объединение усилий.

Манифест Agile

«Лично я не ожидал, что эта конкретная группа [людей] сможет договориться о чем-либо по существу», – позже говорил Алистер Кокберн.

И действительно, за два дня они смогли договориться только о двух вещах: названии Agile и заявлении о четырех ценностях новой методологии (рис. 1.1). В течение следующих нескольких месяцев, переписываясь по электронной почте, эти люди сформулировали 12 сопутствующих принципов (рис. 1.2) [Beck2001].



Манифест гибкой разработки программного обеспечения Agile

Мы постоянно открываем для себя более совершенные методы разработки программного обеспечения, занимаясь разработкой непосредственно и помогая в этом другим. Благодаря проделанной работе мы смогли осознать, что:

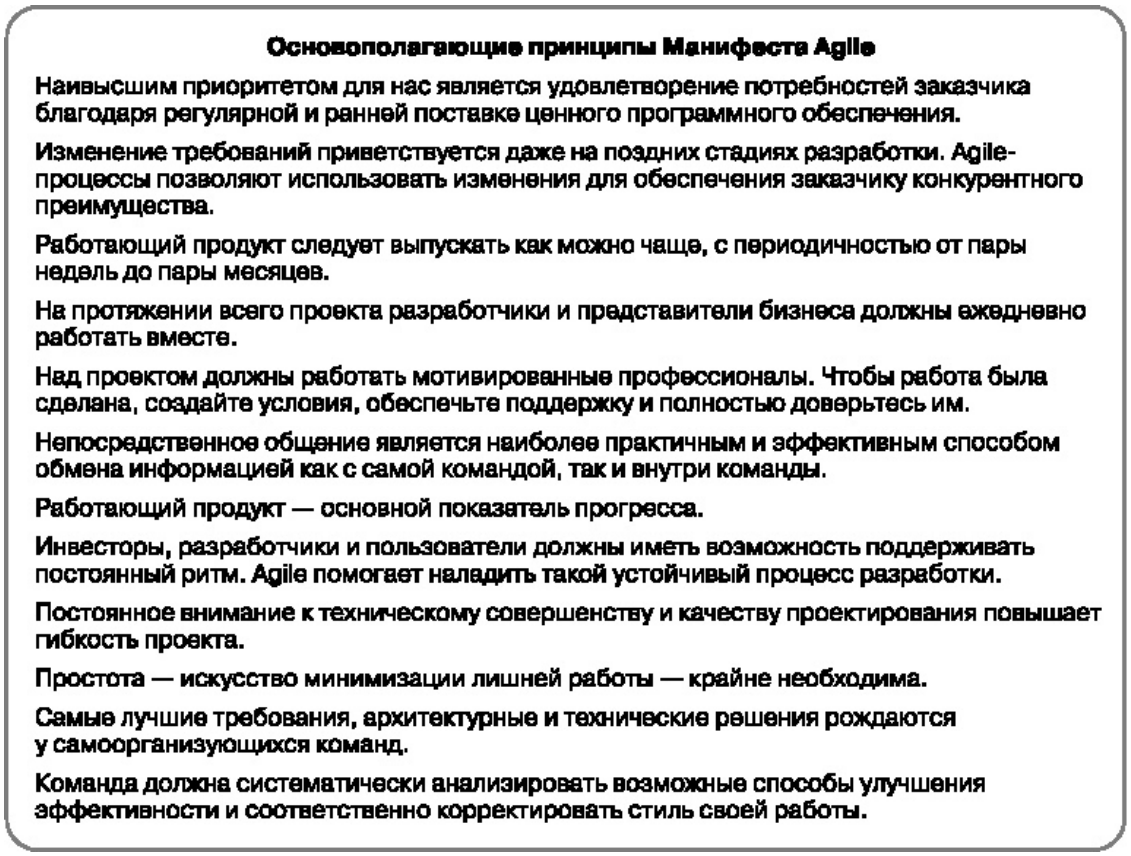
— люди и взаимодействие важнее процессов и инструментов;
— работающий продукт важнее исчерпывающей документации;
— сотрудничество с заказчиком важнее согласования условий контракта;
— готовность к изменениям важнее следования первоначальному плану.

То есть, не отрицая важности того, что справа, мы все-таки больше ценим то, что слева.

Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, Dave Thomas

© 2001, вышеперечисленные авторы
Текст Манифеста может свободно копироваться в любой форме, но только полностью, включая это уведомление.

Рис. 1.1. Ценности Agile



Основополагающие принципы Манифеста Agile

Наивысшим приоритетом для нас является удовлетворение потребностей заказчика благодаря регулярной и ранней поставке ценного программного обеспечения.

Изменение требований приветствуется даже на поздних стадиях разработки. Agile-процессы позволяют использовать изменения для обеспечения заказчику конкурентного преимущества.

Работающий продукт следует выпускать как можно чаще, с периодичностью от пары недель до пары месяцев.

На протяжении всего проекта разработчики и представители бизнеса должны ежедневно работать вместе.

Над проектом должны работать мотивированные профессионалы. Чтобы работа была сделана, создайте условия, обеспечьте поддержку и полностью доверьтесь им.

Непосредственное общение является наиболее практичным и эффективным способом обмена информацией как с самой командой, так и внутри команды.

Работающий продукт — основной показатель прогресса.

Инвесторы, разработчики и пользователи должны иметь возможность поддерживать постоянный ритм. Agile помогает наладить такой устойчивый процесс разработки.

Постоянное внимание к техническому совершенству и качеству проектирования повышает гибкость проекта.

Простота — искусство минимизации лишней работы — крайне необходима.

Самые лучшие требования, архитектурные и технические решения рождаются у самоорганизующихся команд.

Команда должна систематически анализировать возможные способы улучшения эффективности и соответственно корректировать стиль своей работы.

Рис. 1.2. Принципы Agile

Это стало Манифестом Agile, который изменил мир. «Таким образом, – продолжил Алистер, – они в конце концов все же смогли договориться о чем-то по существу»[5].

[5] Алистер Кокберн, цитата Джима Хайсмита в [Highsmith2001]. Полная цитата: «Лично я не ожидал… что эта конкретная группа приверженцев различных гибких методик сможет договориться о чем-либо по существу… Что касается меня, я в восторге от окончательной формулировки [Манифеста]. Я был удивлен, что и другие оказались так же довольны окончательной формулировкой. Таким образом, мы все же смогли договориться о чем-то по существу».

Однако единого метода Agile не было. Его никогда не было и никогда не будет. Agile подразумевает три составляющие: название, ценности и принципы. И все. Это не что-то, что можно делать. Это философия. Это способ мышления, посвященный разработке программного обеспечения. Невозможно «использовать» Agile или «делать» Agile… вы только можете *быть* Agile. Или не быть. Если ваши команды воплощают философию Agile, то вы – Agile. Если не воплощают – то нет.

Суть Agile

Мартин Фаулер сделал карьеру, превращая сложные вопросы о программировании в тщательно продуманные, беспристрастно точные объяснения. Его определение «Суть разработки программного обеспечения по Agile» – одно из лучших:

> Agile-разработка является скорее *адаптивной*, чем предиктивной; *ориентированной* скорее *на людей*, чем на процессы[6].
> *Мартин Фаулер*

Адаптивность вместо предиктивности

Помните, в CHAOS Report утверждалось, что всего одна шестая часть проектов в области программного обеспечения успешна? В отчете успешность определена очень специфическим образом:

• *успешный* – проект выполнен вовремя и в рамках бюджета, все характеристики и функциональности соответствуют изначально запланированным;

сомнительный – проект завершен, результаты переданы в эксплуатацию, но количество характеристик и функциональностей меньше, чем было заявлено изначально. Запланированные бюджет и сроки проекта превышены;

неуспешный – проект отменен в какой-либо момент в течение цикла разработки.

Эти определения прекрасно иллюстрируют предиктивный тип мышления. Они все говорят о *соответствии запланированному*. Если вы сделали то, что собирались сделать, то вы успешны. А если нет, то неуспешны! Все просто.

На первый взгляд, это разумно. Но посмотрим повнимательнее. Тут чего-то не хватает. Райан Нельсон писал в журнале *CIO Magazine* [Nelson2006]:

> Как обнаружилось, проекты, отвечающие всем традиционным критериям успешности – время, бюджет и технические спецификации, – могут по завершении все же быть провальными, поскольку оказываются неактуальными для пользователей, которым предназначались, или потому, что в итоге не приносят особых выгод бизнесу… С другой стороны, проекты, считающиеся неуспешными согласно традиционным метрикам информационных технологий, могут оказаться успешными, поскольку, несмотря на проблемы с бюджетом, графиком или техническими характеристиками, получившаяся система нравится конечным пользователям или имеет какую-либо непредвиденную ценность.

Команды Agile определяют успех как *поставку ценности*, а не соответствие плану. Фактически команды, действительно достигнувшие Agile, активно ищут возможности повысить ценность продукта, *изменив* планы.

---

[6] Фаулер выражал эту идею множество раз в течение нескольких лет. Оригинальную версию можно найти в статье [Fowler2000].

Взгляните еще раз на Манифест (см. рис. 1.1 и 1.2). Найдите пару минут, чтобы вдумчиво изучить ценности и принципы Agile. Сколько из них относятся к поставкам ценного программного обеспечения и адаптации к полученной обратной связи?

Ориентированность на людей, а не на процессы

В случае тяжеловесных процессов люди пытались избежать ошибок, тщательно определяя каждый аспект разработки ПО. При добавлении в процессы регламента индивидуальные навыки становились менее важными. В теории вы могли применять один и тот же процесс снова и снова с разными людьми и получать одни и те же результаты. (Если вдуматься, то они и получали. Просто не те результаты, которые хотели.)

Agile заявляет, что люди – главный фактор успеха в разработке программного обеспечения. Не только их знания и навыки, но и все аспекты человеческой природы. Насколько хорошо сработались члены команды. Насколько часто они отвлекаются. Насколько комфортно и безопасно для них высказывать свое мнение. Насколько они увлечены своей работой.

У Agile-команд тоже есть процессы (у любой команды они есть, пусть и неявные), но они находятся на службе у человека, а не наоборот. И Agile-команды сами отвечают за свои процессы. Желая улучшить методы работы, люди меняют процессы.

Посмотрите на Манифест еще раз (см. рис. 1.1 и 1.2). Какие ценности и принципы имеют отношение к концепции «люди на первом месте»?

Почему Agile победил

В течение первых десяти лет после появления Манифеста Agile столкнулся с небывалой критикой. Его называли «недисциплинированным». Говорили, что он никогда не будет работать. Следующие десять лет критики молчали. Agile был уже везде, по крайней мере это название. Тяжеловесные водопадные методы практически умерли. Молодые программисты вообще не верили в то, что кто-то когда-то мог работать таким образом.

Не то чтобы основанные на фазах процессы неполноценны по сути. У них, безусловно, есть свои недостатки, но если ограничивать их объем, при этом действуя в хорошо изученной предметной области, то методы водопадного стиля тоже могут работать. Проблема была в самом тяжеловесном процессе, который применяли крупные компании. Словно по иронии судьбы, процессы, предназначенные для того, чтобы *избегать* проблем, на самом деле сами *вызывали* большинство проблем, с которыми сталкивались компании.

Трудно представить, как будет работать программа, до того, как вы начнете ее использовать на практике, и еще тяжелее продумать абсолютно все, что она должна будет делать. И это вдвойне сложнее для людей, которые не вовлечены в разработку программного обеспечения. В результате критически важно как можно скорее предоставить людям работающую программу. Вам просто необходимо получить от них обратную связь о том, что не работает и чего не хватает, и затем скорректировать ваши планы в зависимости от полученной информации. Как говорится в Манифесте, «работающее программное обеспечение – основной показатель прогресса». Получение информации и реакция на изменения лежат в основе всего того, что называется Agile.

В случае тяжеловесных процессов придавалось настолько большое значение контролю над процессами и согласованию документации, что порождались значительные задержки и расходы. На то, чтобы получить работающее ПО, уходили годы, и заказчику не показывали ничего конкретного почти до самого конца проекта. Вместо того чтобы приветствовать изменения, в этих процессах делалось все, чтобы их *избежать*. Была даже отдельная составная часть про-

цессов «Совет по контролю за изменениями» (Change Control Board), чьей основной задачей было сказать «нет» запросам на изменения. (Точнее, «да, но за это понадобится доплатить».)

Все это наслаивалось друг на друга в проектах, где люди годами вели разработку, прежде чем могли что-то показать клиенту. И когда они это делали, было уже слишком поздно и дорого что-то менять. В конечном итоге они выдавали программное обеспечение, которое не делало то, чего хотел заказчик.

Типичный провал тяжеловесного процесса

Третьего февраля 2005 года Роберт С. Мюллер III, директор Федерального бюро расследований, предстал перед подкомитетом Сената США, чтобы объяснить, как ФБР умудрилось потратить впустую 104,5 миллиона долларов[7].

Это явно было не комфортно для компании. В июне 2001-го ФБР запустило проект VCF с целью заменить программное обеспечение для управления делами. Через четыре года он был отменен. Общие затраты составили 170 миллионов долларов; 104,5 миллиона из них были полностью невозвратными.

Сроки и последовательность событий проекта могут рассказать нам хорошо знакомую историю. Проект начался в июне 2001 года. Через 17 месяцев, в ноябре 2002-го, были определены «четкие требования». Программное обеспечение было поставлено год спустя, в декабре 2003-го, но «ФБР сразу обнаружило некоторое количество недостатков в VCF, которые делали его непригодным для использования». Подрядчик, разрабатывавший программу, согласился исправить недостатки, но только за дополнительную плату в размере 56 миллионов долларов и в срок один год. В конце концов ФБР отказалось от идеи исправлять недочеты, фактически перечеркнув годы работы.

Несмотря на существование большого разнообразия подходов к Agile (и некоторые из них больше используют популярное название, чем действительно следуют философии), у них есть кое-что общее. Они фокусируются на том, чтобы делать прогресс в работе видимым, позволяя всем стейкхолдерам проекта корректировать его на ходу. Казалось бы, это мелочь, но она невероятно эффективна. Именно благодаря ей мы больше не слышим о кризисе ПО. Сроки разработки программ все еще задерживаются. Бюджеты все еще перерасходуются. *Но Agile-команды демонстрируют прогресс, используя работающие программы, а не документы.* С самого начала. И это огромное достижение.

Agile имеет и другие преимущества, помимо наглядности процесса. Но даже одного этого оказалось достаточно, чтобы все захотели стать Agile.

Почему Agile работает

Первым прорывным успехом Agile стало экстремальное программирование, метод со слоганом «Примите изменения». В нем смешались здоровая доза философских размышлений о разработке программ и прагматичное стремление изменить ситуацию к лучшему. В предисловии к первой книге по экстремальному программированию было сказано:

> «Если коротко, то экстремальное программирование обещает снизить риски проекта, улучшить отклик на потребности бизнеса и повысить

[7] Источники: Mueller's February 3, 2005, testimony to Congress (https://oreil.ly/GlQSa) и Inspector General Glenn Fine's May 2, 2005, testimony to Congress (https://oig.justice.gov/node/672).

> продуктивность в течение всего срока жизни системы, сделать приятным создание ПО в команде – и все это одновременно. Это правда. Прекратите смеяться» [Beck2000a].
> *Extreme Programming Explained, первое издание*

Одни действительно смеялись. А другие попробовали и обнаружили, что (вопреки всеобщему мнению о том, как должна вестись разработка ПО) экстремальное программирование реально выполняло все, что обещало. Так, несмотря на смех, оно пользовалось спросом, а вместе с ним Agile.

Экстремальное программирование было живым примером Agile, заложившим основу идей и терминологии, которые используются до сих пор. Однако сила сообщества Agile в том, что оно всегда было широкой коалицией. Agile не ограничивается каким-то одним методом. Он постоянно расширяется, включая в себя новых людей и свежие идеи. «Бережливая разработка программного обеспечения» (Lean software development), Scrum, Kanban, «Бережливый стартап» (Lean Startup), DevOps и многие-многие другие подходы внесли свой вклад в то, что сейчас известно под общим названием Agile.

Если взять все эти идеи и сгруппировать по категориям, то получится пять центральных концепций.

• *Полагайтесь на людей.* Создавайте процессы, которые понятны и могут работать с учетом человеческой природы. Отдайте право принимать решения в руки тех, кто наиболее квалифицирован в этой области. Выстраивайте здоровые рабочие взаимоотношения, основанные на сотрудничестве.

Поставляйте полезный продукт. Запрашивайте обратную связь, экспериментируйте и подстраивайте свои планы. Считайте частично выполненную работу затратами, а не прибылью. Концентрируйтесь на создании ценных результатов. Поставляйте продукт часто.

Исключите напрасную трату времени и усилий. Выполняйте работу маленькими обратимыми шагами. Примите возможность неудачи и стройте ваши планы с учетом вероятности их быстрого провала. Максимизируйте невыполненную работу. Стремитесь к производительности, а не эффективности.

Стремитесь к техническому совершенству. Обеспечивайте гибкость с помощью технического качества. Закладывайте в проект то, что известно, а не то, что предполагаете. Начинайте с простого и добавляйте сложное, только если это будет реально необходимо. Создавайте системы, которые легко развивать, даже (и особенно) в непредвиденных направлениях.

• *Совершенствуйте ваши процессы.* Экспериментируйте с новыми идеями. Настраивайте и адаптируйте то, что работает. Никогда не считайте, что отлаженный, популярный способ будет самым лучшим и для вас.

Agile определяется Манифестом, но сам Манифест – лишь начальная точка. Agile работает потому, что люди *заставляют* его работать. Они берут идеи Agile, адаптируют их к своим ситуациям и не перестают совершенствоваться.

Почему Agile терпит неудачу

Agile начинался как общественное движение. Первоначально он был успешным в значительной степени благодаря программистам, стремящимся к лучшим результатам и лучшему качеству жизни. По мере роста этого успеха акцент сместился с основополагающих идей в сторону ажиотажа. Вместо того чтобы сказать: «Давайте добьемся лучших результатов, адаптируя наши планы и ставя интересы людей превыше всего», – лидеры компаний стали говорить: «Все обсуждают Agile. Дайте мне этот Agile».

Дело в том, что нет такого Agile, который можно было бы где-то взять. Это просто набор идей. Существуют специфические подходы Agile, такие как экстремальное программирование

и Scrum, которые могут вам объяснить, *как* быть Agile, но вам все же придется принять лежащую в его основе философию.

А для многих организаций эта основополагающая философия (адаптировать планы и ставить интересы людей превыше всего) на самом деле *реально* чуждая.

Карго-культы

Корни этой истории уходят в 1940-е[8], когда американские войска высадились на далеком острове. Местные жители никогда не встречались с современной цивилизацией, и люди и материалы, привезенные союзными войсками, изумили аборигенов. Они наблюдали, как солдаты строили взлетно-посадочную полосу и башню управления воздушным трафиком, надевали наушники и вызывали с небес на землю больших металлических птиц, нагруженных ценным грузом (карго). Когда птицы приземлялись, часть груза распределялась между всеми островитянами, благодаря чему те жили в благополучии и комфорте.

Однажды войска покинули остров, и большие металлические птицы перестали прилетать. Успевшие привыкнуть к комфорту островитяне сплели из бамбука собственную взлетно-посадочную полосу. Они построили высокую платформу, посадили на нее своего вождя и надели ему на голову кокосы, которым придали форму наушников. Но как бы ни старались островитяне, большие металлические птицы так больше никогда и не вернулись.

Трагедия карго-культа – в его приверженности к поверхностным, внешним проявлениям какой-либо идеи в сочетании с полным непониманием того, как эта идея работает на самом деле. В той истории островитяне воссоздали все элементы по отдельности: взлетно-посадочную полосу, башню, наушники, – но не знали обо всей инфраструктуре, благодаря которой прибывали самолеты.

Такая же трагедия случается с Agile. Люди хотят Agile-карго: лучшие результаты, больше наглядности происходящего процесса, меньше неудач бизнеса. Но они не понимают лежащую в основе всего этого философию и зачастую не согласились бы с ней, даже если бы поняли. Они хотят купить Agile, но идею нельзя купить.

Что *можно* купить, так это внешние символы Agile. Стендап-митинги! Истории! Инструменты! Сертификации! Есть множество вещей с маркировкой Agile и достаточно людей, жаждущих вам их продать. Их часто продают как элементы «корпоративного уровня», что является способом сказать: «Не беспокойтесь, вам не придется ничего менять». Неудобные идеи, вроде «адаптивного планирования» или «ориентированности на людей», игнорируются.

Так и получается карго-культ. Много действий, ноль результата. Относящаяся к Agile составляющая отсутствует.

«В моей предыдущей компании огромное количество человеко-часов тратилось на совещания».

«Из-за этого [Agile] целая команда (30+ человек) лишилась работы, поскольку они не произвели почти ничего в результате почти года работы».

«Все это [Agile] означает, что разработчиков просто обманули, когда проект изменился… за день до поставки».

Реальные комментарии об Agile из сети

[8] Впервые я прочел эту историю в трудах Ричарда Фейнмана, где в числе прочего была представлена его речь на церемонии вручения дипломов в Калтехе в 1974 году [Feynman1974]. История основана на реальных ритуалах, практиковавшихся в Меланезии после Второй мировой войны.

Слово Agile – повсюду. *Идеи* Agile – нет. Это вечный самообман для многих: единственный Agile, который они знают, – это карго-культ Agile.

Настало время это исправить. В этой книге я расскажу вам, как применять идеи Agile по-настоящему. Будьте начеку, чтобы не поддаться поборникам карго-культа Agile, которых вы встретите в книге. Они покажут вам, как делать *не* надо.

Готовы? Поехали.

Глава 2. Как быть Agile

Как перейти от нагромождения Agile-идей к реальным, действующим Agile-командам? Нужна практика. Очень много практики.

Практика Agile

У каждой команды есть свой подход к работе – *процессы*, или *методы*, которым она следует, зачастую формально не задокументированы. Эти методы отражают лежащую в их основе философию разработки программного обеспечения, хотя она не всегда бывает четко сформулирована и не обязательно последовательна.

Чтобы быть Agile, вам нужно изменить свои методы так, чтобы они стали отражать философию Agile. Это одновременно и проще, и сложнее, чем кажется. Это просто, поскольку в большинстве случаев вы можете начать с одного из готовых Agile-методов, например, представленного в этой книге. И это сложно, так как вам понадобится перестроить свой стиль работы, а это подразумевает изменение многих привычек.

Сообщество Agile называет эти привычки *практиками*. Им посвящена значительная часть данной книги. Под практиками подразумеваются сессии планирования, автоматизированные сборки и демо для стейкхолдеров. Большинство этих практик существует десятилетиями. В методах Agile они скомбинированы уникальным образом – выделяются компоненты, поддерживающие философию Agile, отбрасываются остальные и добавляется ряд новых идей. В результате получается экономичный, эффективный, взаимоусиливающий комплект.

Практики Agile часто имеют двойное и тройное назначение, решая одновременно множество проблем и поддерживая друг друга разумными и неожиданными способами. Вам не удастся по-настоящему понять, как работает метод Agile, до тех пор, пока вы некоторое время не понаблюдаете за ним в действии.

Таким образом, несмотря на соблазн сразу настроить метод Agile под себя, лучше все же начать со стандартного, «книжного» подхода. Идея удалить наименее знакомые практики выглядит заманчиво, но именно они и будут вам нужны *больше всего*, если вы действительно собираетесь перейти на Agile. Именно с этими практиками связаны главные изменения в философии.

Как достичь мастерства

Овладение искусством разработки по Agile требует реального опыта использования конкретного, четко определенного метода Agile. Начните со стандартного, «книжного» подхода. Примените его на практике – *полностью* – и потратьте несколько месяцев, совершенствуя практику и досконально разбираясь, почему она работает. *Потом* настройте под себя. Выберите один из нюансов, разберитесь с последовавшими переменами и повторите.

Данная книга посвящена этой цели. В ней представлен тщательно подготовленный набор практик Agile, проверенных в реальном мире. Чтобы совершенствоваться в искусстве Agile или просто достичь бóльшего успеха с помощью практик Agile, выполняйте следующие шаги.

1. Выберите для освоения подгруппу идей Agile. Глава 3 поможет вам определиться.

2. Применяйте как можно больше соответствующих практик. Они описаны в частях II–IV. Практики Agile усиливают друг друга, поэтому работают наилучшим образом, если использовать их все вместе.

3. Применяйте практики строго и последовательно. Если практика не работает, то попробуйте следовать методу *еще точнее*. Команды-новички в Agile часто недостаточно тщательно

следуют практикам. Будьте готовы к тому, что вам потребуются два-три месяца на то, чтобы почувствовать себя комфортно, работая с новыми практиками, и еще от двух до шести месяцев на то, чтобы довести их использование до автоматизма.

4. По мере того как вы почувствуете уверенность в том, что применяете практики правильно (повторимся, отведите на это несколько месяцев), начните экспериментировать, внося изменения. Описание каждой из практик в этой книге дополнено рассуждениями о том, почему она работает и что в ней можно изменить. Каждый раз, меняя что-либо, наблюдайте за результатами и затем вносите дальнейшие улучшения.

5. Последнего шага нет. Разработка программного обеспечения по Agile – это непрерывный процесс обучения и совершенствования. Не прекращайте практиковаться, экспериментировать и развиваться.

На рис. 2.1 показан этот процесс. Сначала следуйте правилам, потом нарушайте их и в конце концов будьте выше правил[9].

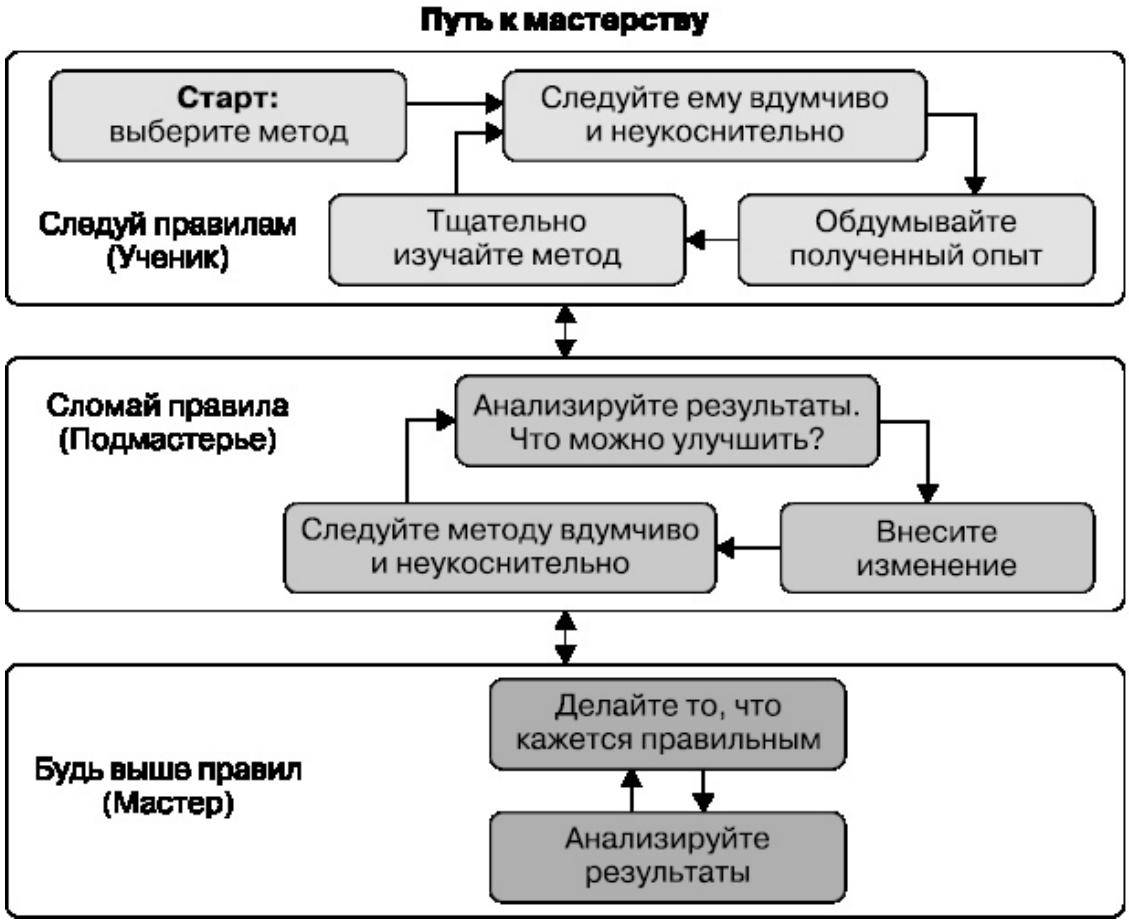


Рис. 2.1. Как достичь мастерства

С чего начать?

Ваши первые шаги зависят от того, чего вы хотите. Вы присоединяетесь к действующей команде Agile, внедряете Agile в работу одной или нескольких команд или стремитесь улучшить свою команду Agile?

[9] Эта последовательность действий следует из рассуждений Алистера Кокберна о сюхари (Shu-Ha-Ri).

Присоединение к действующей команде Agile

Если вы планируете присоединиться к действующей Agile-команде или просто хотите дополнить свои знания о том, как Agile работает на практике, то можете перейти к частям II–IV. Каждая часть начинается с истории в стиле «один день из жизни», описывающей, как может выглядеть Agile. Команды, работающие по Agile, отличаются друг от друга, поэтому и ваша не будет абсолютно такой же. Но эти истории дадут вам представление, чего можно ожидать.

Прочитав истории, переходите к интересующим вас практикам. Каждая написана как отдельный справочный материал.

Введение Agile

Если вы помогаете своей организации сформировать Agile-команды или только хотите убедить ее сделать это, то оставшиеся главы части I помогут вам начать. Чтобы структурировать процесс, используйте чек-лист, представленный ниже.

Сначала убедитесь, что Agile подходит вашей компании:

• выберите подход Agile, который ваша организация сможет поддерживать (см. главу 3);

• определите, что ей нужно сделать, чтобы успешно внедрить Agile (см. главу 4);

• получите согласие на эксперимент с Agile (см. главу 5);

• если у вас несколько команд, то решите, как вы будете их масштабировать (см. главу 6).

За несколько недель до начала:

• определитесь, кто будет коучем (или коучами) команды, и выберите по меньшей мере одного человека, который будет продакт-менеджером (см. раздел «Вся команда» главы 7);

• организуйте встречу продакт-менеджера с исполнительным спонсором команды и ключевыми стейкхолдерами, чтобы разработать замысел проекта (см. раздел «Цель» главы 7);

• обеспечьте команде физическое или виртуальное помещение (см. раздел «Командная комната» главы 7);

• запланируйте и проведите сессию подготовки устава команды (см. врезку «Планирование сессии подготовки устава» в главе 7);

• попросите команду ознакомиться с новым методом. Раздайте людям копии этой книги, чтобы они могли ее изучить самостоятельно, предложите применить некоторые практики в их текущей работе и рассмотрите возможность проведения тренинга по тем практикам, которые вызывают трудности (практики представлены в частях II–IV).

Когда команда будет готова начать, сделайте глубокий вдох и:

• поручите членам команды спланировать первую рабочую неделю (см. подраздел «Ваша первая неделя» главы 9).

Совершенствование действующих Agile-команд

Если у вас уже есть действующие Agile-команды и вы хотите улучшить их работу, то ваш подход будет зависеть от того, какие именно улучшения вам нужны.

Если вы заинтересованы в небольшой регулировке уже работающих процессов вашей команды, то перейдите к частям II–IV и прочитайте об интересующих вас практиках. Если вы хотите более масштабных улучшений, то процесс будет таким же, как и при внедрении Agile в команду, за исключением того, что вам понадобится сосредочиться на конкретных элементах, требующих изменений.

В качестве руководства используйте чек-листы из предыдущего подраздела «Внедрение Agile» данной главы.

Если Agile не срабатывает в вашей организации, то ознакомьтесь с врезкой «Руководство по устранению неполадок» в главе 4.

Применение отдельных практик Agile

Agile работает наилучшим образом, когда вы идете ва-банк, но если это не ваш вариант, то вы можете добавить немного Agile в действующие процессы. Вот подходящие практики, с которых можно начать.

• *Ежедневное планирование*. Если вы боретесь с частыми прерываниями (interruptions), то попробуйте использовать однодневные итерации (см. раздел «Планирование задач» главы 9). Возьмите за основу игру в планирование (см. раздел «Игра в планирование» главы 8) и измеряемый потенциал (capacity) по работе вашей команды на спринте (см. раздел «Потенциал» главы 9) и проводите совместные сессии планирования в начале каждого дня, откладывая все прерывания на сессию планирования следующего дня. Обеспечьте условия, чтобы люди сами оценивали свои задачи.

Итерации. Если частые прерывания для вас не проблема, но вы все же хотели бы улучшить свое планирование, то попробуйте использовать недельные итерации (см. раздел «Планирование задач» главы 9). В этом случае вы также можете практиковать ежедневные рабочие стендап-митинги (см. раздел «Стендап-митинги» главы 9) и регулярные демо для стейкхолдеров (см. раздел «Демо для стейкхолдеров» главы 10). Со временем рассмотрите возможность использования индексных карточек для планирования и больших диаграмм, показывающих предстоящую работу, как описано в разделе «Визуальное планирование» главы 8.

Ретроспективы. Частое проведение ретроспектив (см. раздел «Ретроспективы» главы 11) – отличный способ адаптировать и улучшить рабочие процессы команды. Могут быть полезны и другие практики, перечисленные в главе 11.

Быстрая обратная связь. Быстрая автоматизированная сборка значительно улучшит вашу жизнь, а также откроет возможности для других усовершенствований (см. раздел «Нулевое трение» главы 13).

Непрерывная интеграция. Непрерывная интеграция (практика, а не инструмент) не только уменьшает проблемы интеграции, но и способствует повышению качества сборок и тестов. Более подробную информацию см. в разделе «Непрерывная интеграция» главы 13.

• *Разработка через тестирование*. Хотя эту практику не так легко принять, как другие, она весьма эффективна. Разработка через тестирование (см. соответствующий раздел главы 13) позволяет снизить частоту появления программных ошибок (багов), повысить скорость разработки, улучшить вашу способность к переработке кода (рефакторингу) и сократить технический долг. На ее освоение может уйти некоторое время, так что запаситесь терпением.

> **Другие практики, описанные в частях II–IV, также могут оказаться полезными. Agile-практики объединены множеством зависимостей друг от друга, поэтому обязательно обратите внимание на блоки «См. также» и подраздел «Предварительные требования» каждой практики.**

Не расстраивайтесь, если возникнут проблемы с применением отдельных практик. Быстрее и проще выбрать соответствующую группу практик и применить ее полностью, от начала и до конца. Это мы и рассмотрим далее.

Глава 3. Выберите свою гибкость

Нет смысла использовать Agile ради него самого. Задайте себе два вопроса.
1. Поможет ли нам Agile стать более успешными?
2. Чего нам будет стоить достижение этого успеха?
Ответив на эти вопросы, вы поймете, нужен ли вам Agile.

Что ценно для организаций?

Успех определяется не только доходами. Вот только несколько составляющих успеха:

• *улучшение финансовых результатов* – прибыль, рост выручки, биржевая стоимость акций, снижение затрат;

• *достижение целей организации* – стратегические цели, оригинальные исследования, благотворительные цели;

• *укрепление позиций на рынке* – продвижение бренда, конкурентные различия, лояльность существующих клиентов, привлечение новых;

• *обретение популярности* – стратегическая информация, аналитика, отзывы клиентов;

• *снижение риска* – безопасность, требования законодательства, аудит;

• *увеличение потенциала* – наем и удержание персонала, моральный климат и развитие навыков, автоматизация.

Модель Agile Fluency

В 2014 году я сотрудничал с Дианой Ларсен, чтобы проанализировать, почему компании видят разные результаты от их Agile-команд. Мы оба работали с командами с самого начала. С годами мы заметили, что команды начинали склоняться к кардинально разным типам результатов и эти результаты имели тенденцию группироваться в разных «зонах».

Мы обобщили эти наблюдения в модели Agile Fluency™. Ее упрощенный вариант показан на рис. 3.1 [Shore2018b].

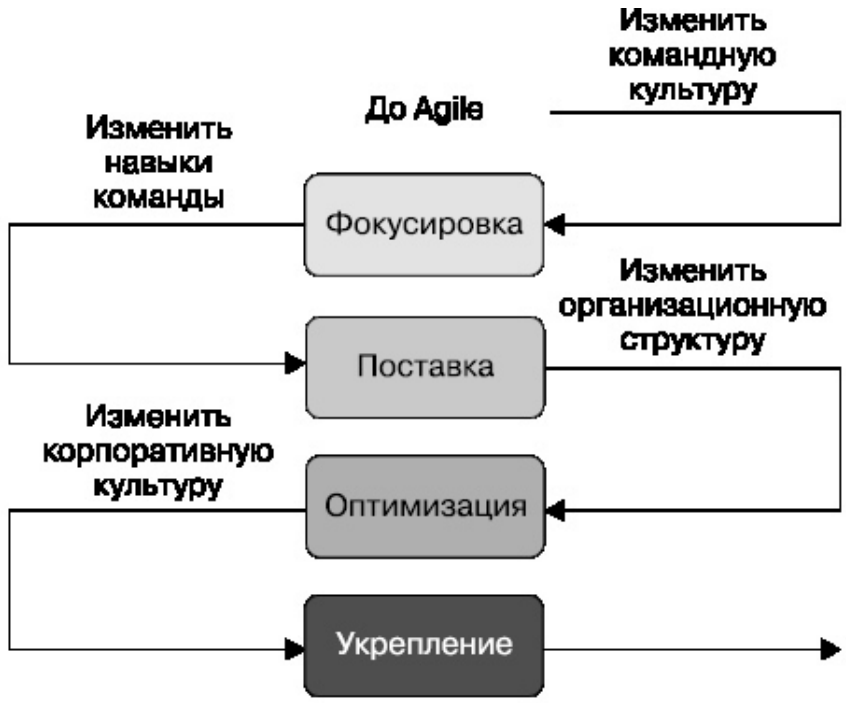


Рис. 3.1. Упрощенная модель Agile Fluency™

Каждый уровень связан с набором преимуществ. Чтобы обрести их, команде необходимо свободно (уверенно) владеть навыками, относящимися к этому уровню. Считается, что команда свободно владеет навыками, если способна применять все навыки этого уровня, не прикладывая сознательных усилий.

ПРИМЕЧАНИЕ

Несмотря на то что на рис. 3.1 представлен прямой путь от одного уровня к другому, в реальности все более беспорядочно. Команды могут достигать уверенного владения навыками на любом уровне, в любом порядке, хотя продвижение, показанное на рис. 3.1, типично.

Навыки, необходимые для свободного применения, перечислены в предисловиях к частям II–IV. Но уверенное владение навыками не достигается само по себе. Ваша *организация* тоже должна инвестировать в эти навыки своих команд. Это гораздо больше, чем поддерживать идеи Agile на словах. Организация должна произвести реальные, значительные изменения, которые занимают время, стоят денег и требуют вложения политического капитала.

Результат, который вы получаете от команд, зависит от того, насколько ваша организация вкладывается в идеи Agile. Компании не удается добиться результатов, которых она хочет от Agile, обычно потому, что она не сделала необходимые инвестиции. Часто организации даже не осознают, что это было нужно.

Инвестирование в Agile должно стать осознанным выбором. Тщательно изучите каждый из уровней. Каждый требует затрат, и каждый имеет свои преимущества. Выберите те уровни, которые имеют наилучшее соотношение затрат и выгоды для вашей ситуации.

Вероятнее всего, вам не удастся убедить вашу компанию инвестировать в каждый уровень. Это нормально. В отличие от моделей зрелости (maturity models), таких как интегрированная модель зрелости возможностей компании при разработке ПО (Capability Maturity

Model Integration, CMMI), модель Agile Fluency™ не показывает продвижения от слабых навыков к сильным. Вместо этого она демонстрирует множество *вариантов* соотношения инвестиций/выгоды. Хотя диаграмма показывает наиболее распространенный вариант продвижения, каждый уровень можно выбрать независимо. Каждый имеет ценность сам по себе.

Уровень владения навыками и степени зрелости

Свободное владение навыками – это свойство, присущее *команде*, а не одному индивидууму. Свободное владение навыками не означает, что каждый член команды обладает каждым навыком, связанным с данным уровнем. Вместо этого им нужна способность, будучи командой, привлекать к работе правильных людей в правильное время.

Каждый уровень имеет несколько степеней зрелости.

1. Обучение – команда изучает нужные навыки.

2. Профессионализм – команда может продемонстрировать нужные навыки, когда она концентрируется на них.

3. Свободное владение – команда демонстрирует нужные навыки автоматически, не прикладывая осознанных усилий, при условии, что в команде есть коуч.

4. Самостоятельное свободное владение – команда демонстрирует нужные навыки автоматически, не нуждаясь в присутствии в команде коуча.

Уровень фокусировки (Focusing)

Уровень *фокусировки* – это самые основы Agile: сфокусированность на бизнес-результатах; командная работа; принятие на себя ответственности. Команды, которые свободно владеют навыками на этом уровне, фокусируют процесс разработки на основной цели команды, сначала выпускают наиболее ценные функциональности и меняют направление в ответ на меняющиеся потребности бизнеса. Они постоянно *фокусируются* на наиболее значимых приоритетах своей организации.

В большинстве команд и организаций для этого требуется изменить образ мышления о командах. Организации, находящиеся в состоянии «до Agile», составляют планы заранее, запрашивают смету у команды и требуют отчетов о соответствии хода работ этой смете. Команды зоны *фокусировки* часто корректируют свои планы (по меньшей мере ежемесячно) и демонстрируют прогресс, показывая выполненную работу.

Организации в состоянии «до Agile» разбивают свои планы на задачи, назначают членов команды ответственными за них и оценивают людей, основываясь на том, насколько хорошо те выполняют свои задачи. Команды уровня *фокусировки* самостоятельно делают разбивку задач, сами решают, кто над чем будет работать, и ожидают, что их будут оценивать по их способности создавать ценный продукт как команда.

Чтобы команды вашей организации достигали успеха, понадобится поддерживать перемены конкретными инвестициями в форме изменения командной структуры, системы управления и рабочей среды. (Я подробно расскажу об этом в следующей главе.) Это ситуация в стиле «хорошие новости и плохие новости»: плохие новости в том, что когда доходит до дела, у многих организаций пропадает желание инвестировать. А хорошие новости таковы: если они отказываются, то вы понимаете сразу, на начальном этапе, что *на самом деле* они не готовы к погружению в философию Agile. Вы просто избавили себя от долгих лет разочарований и душевной боли, которые пришлось бы испытать, следуя за карго-культом Agile.

Если же вам удастся получить поддержку, то владение навыками на уровне *фокусировки* может быть достигнуто каждой командой примерно за 2–6 месяцев целенаправленных усилий.

При должной поддержке они превзойдут свой предыдущий уровень производительности даже в течение 1–4 месяцев[10]. В части II приведены практики, которые могут им понадобиться.

Уровень поставки (Delivering)

Agile-команды могут менять свои планы в любое время. Для большинства команд это несколько снижает качество их кода. Они постепенно теряют свою способность вносить изменения, эффективные с точки зрения затрат. В итоге команды могут сказать, что нужно выбросить ПО и переписать его заново, а это дорогое и невыгодное решение.

Команды уровня *поставки* предотвращают эту проблему через свое техническое превосходство. Они сразу закладывают в дизайн своих программ готовность к частым изменениям. Команды поддерживают высокое качество кода, поэтому не тратят время на поиски багов. Они совершенствуют цикл своего производства так, чтобы релизы ПО становились безболезненными, а эксплуатация – управляемой. Они способны *поставить* надежное программное обеспечение с низким уровнем дефектов в любой момент, когда это наиболее целесообразно для бизнеса.

Достижение таких результатов требует существенных инвестиций в навыки членов команды по разработке, а также структурных изменений, которые позволят интегрировать людей, компетентных в тестировании и эксплуатации, в каждую команду.

Если ваша компания сделает эти инвестиции, то на достижение владения навыками на уровне *поставки* у ваших команд уйдет от 3 до 24 месяцев, и повышение производительности вы сможете увидеть в течение 2–6 месяцев. Точное количество времени, которое понадобится каждой команде, зависит от текущего качества ее кода и от того, сколько у нее будет коучей. Часть III содержит нужные практики.

Уровень оптимизации (Optimizing)

Большинство компаний были бы довольны лишь уровнями *фокусировки* и *поставки*. Но Agile способен на большее. Agile во всем своем великолепии – это мир, в котором команды рады изменениям рыночных условий. Они экспериментируют и учатся, развивают новые рынки, обыгрывают конкурентов.

Команды уровня *оптимизации* достигают этой степени гибкости. Они понимают, чего хочет рынок, каковы требования бизнеса и как можно соединить одно с другим. Или, как в обстановке стартапа, они понимают, чему им надо научиться и как к этому приступить. Они постоянно *оптимизируют* свои планы создания продукта, чтобы те достигли максимально возможного уровня.

Здесь необходимы изменения в организационной структуре. Создание оптимальных планов требует постоянного внимания людей, обладающих глубокими знаниями в области бизнеса и продуктов, и, следовательно, эксперты по рынку и по продукту должны постоянно взаимодействовать с командой разработки. Это также означает, что на эти команды будет возложена полная ответственность за их планы создания продукта и бюджеты.

Такие структурные изменения требуют одобрения на высоком уровне, которое может быть трудно получить. Команды обычно тратят по меньшей мере год на укрепление доверия с помощью уверенного владения навыками на уровне *поставки*, прежде чем получают разрешение на такие инвестиции. Как только это происходит, свободное владение навыками в *оптимизации* занимает следующие 3–9 месяцев, хотя увидеть некоторые улучшения можно уже в течение 1–3 месяцев. Но даже в этом случае *оптимизация* остается бесконечным процессом экспериментирования, обучения и открытий. В части IV мы поговорим о том, как начать.

---

[10] Временные рамки, приведенные в этой главе, приблизительны и основаны на моем опыте. Ваш опыт может быть другим.

Уровень укрепления (Strengthening)

Есть еще один последний уровень в модели Agile Fluency™. Он во многом теоретичен: это возможное будущее Agile. Кроме того, он подходит только для организаций, находящихся на переднем крае теории и практики управления. Это обстоятельство выводит данный уровень за рамки темы этой книги. Вкратце, уровень *укрепления* предполагает переработку коллективных идей команд и направление их в организационные улучшения. Если вы хотите узнать об этом больше, то прочтите главу 19.

Уровни свободного владения навыками Agile, резюме

Фокусировка

• *Основные преимущества:* фокус на бизнес-приоритеты, наглядность работы команды, способность менять направление.

• *Инвестиции:* структура команд, управление, рабочая среда.

• *Примерные сроки:* падение производительности на 1–4 месяца, достижение владения наыками в течение 2–6 месяцев.

Поставка

• *Основные преимущества:* мало дефектов и высокая производительность, техническая долговечность.

• *Инвестиции:* навыки разработки, слияние тестирования и эксплуатации.

• *Примерные сроки:* падение производительности на 2–6 месяцев, достижение владения навыками в течение 3–24 месяцев.

Оптимизация

• *Основные преимущества:* повышение уровня ценности релизов и улучшенные продуктовые решения.

• *Инвестиции:* встроенное управление продуктом, команда несет ответственность за бюджет и планы.

• *Примерные сроки:* падение производительности на 1–3 месяца, достижение владения навыками в течение 3–9 месяцев.

Выберите свои уровни

К каким уровням свободного владения навыками должна стремиться ваша команда? Это зависит от того, на каких из них вас может поддержать ваша организация. В идеале все вместе: *фокусировка, поставка* и *оптимизация* – наилучший вариант. Комбинация трех уровней обеспечивает наилучшие результаты и чистейшую реализацию идей Agile.

Но выбор всех трех уровней одновременно требует и максимальных инвестиций. Если вы не можете обосновать их, то у вас наверняка будут сложности с получением необходимой поддержки. А при недостаточных инвестициях вашим командам будет тяжело достичь уверенного владения навыками. Вы понесете расходы на обучение и не получите преимуществ. Вы даже можете увидеть *худшие* результаты по сравнению с тем, что есть сейчас.

Другими словами, выбирайте те уровни, которые *нужны* вашей компании и за которые она *хочет платить*.

Какие же уровни вам все-таки выбрать?

• Каждая команда нуждается в свободном владении навыками на уровне *фокусировки*. Это базовое свойство. Если ваша компания не может инвестировать по меньшей мере в навыки в *фокусировке*, то, вероятно, Agile вам не подходит, хотя, возможно, у вас получится посте-

пенно продвигаться в этом направлении, если вы начнете со свободного владения навыками на уровне *поставки*.

• Свободное владение навыками на уровне *поставки* снижает затраты и повышает скорость разработки. Без нее ваш код в конечном итоге скатится к техническому долгу. Это делает уровень *поставки* очевидной целью для большинства команд. Тем не менее многие организации не готовы к большим инвестициям в обучение и качество кода, требующимся для уровня *поставки.* Тогда имеет смысл начать со свободного владения навыками на уровне *фокусировки,* продемонстрировать успех и использовать его как положительный пример для обоснования дальнейших инвестиций.

• Свободное владение навыками на уровне *оптимизации* — это то, где Agile проявляет себя наиболее ярко. Но хотеть этого сразу, возможно, чересчур. Для большинства организаций лучше сначала выстроить доверительные отношения, продемонстрировав навыки на уровнях *фокусировки* и *поставки,* а затем постепенно брать на себя больше ответственности. Но если в вашей организации уже принято делегировать полномочия по принятию решений кросс-функциональным командам, как это часто бывает в стартапах, то благодаря свободному владению навыками на уровне *оптимизации* вы получите великолепные результаты.

Больше информации о каждом уровне и его преимуществах вы можете найти в предисловиях к частям II–IV. Подробная информация о необходимых инвестициях представлена во врезке «Список необходимых инвестиций» в главе 4. Если вы не можете определиться, какой из уровней выбрать, то начните с *фокусировки* и *поставки*.

Какие бы уровни вы ни выбрали, инвестируйте в изучение всех относящихся к ним практик одновременно. Практики из более поздних уровней улучшают работу более ранних, поэтому вам лучше осваивать их все вместе, а не по отдельности. Но если вы не можете инвестировать в каждый из желаемых уровней, то это нормально. Процесс будет длиться дольше, но со временем вы сможете добраться и до других уровней.

Определившись с желаемыми уровнями, переходите к более подробному рассмотрению инвестиций, требуемых от вашей организации. Мы будем изучать их в следующей главе.

Глава 4. Инвестируйте в гибкость

Как мы видим из предыдущей главы, для того чтобы команды получили все преимущества Agile, ваша организация должна приобщиться к его основополагающей философии. Не просто тратить деньги (это сравнительно легко), а выполнять реальные, осмысленные изменения в организационных структурах, системах и рабочих привычках…

Если это выглядит как очень трудозатратное дело… что ж, так и есть. Неужели эти инвестиции действительно настолько важны?

Да, действительно настолько.

Инвестировать в Agile важно, поскольку вы инвестируете в изменение *границ, которые вас сдерживают*. По большей части команды сдерживают не процессы, которым они следуют, а ограничения, в которых они находятся. Попробуйте делать эти инвестиции и не следовать практикам, и ваши команды, вероятно, все же покажут улучшение результатов. А если, наоборот, следовать практикам и игнорировать инвестиции? Тогда командам придется тяжело.

Как сказал Мартин Фаулер[11], «я вижу поразительную параллель между ДХХ (Давид Хейнемейер Ханссон, создатель Ruby on Rails) и Кентом Беком (создатель экстремального программирования). Любой из них, получив в подарок ограниченный мир, посмотрит на его ограничения, которые мы принимаем как должное, сочтет их несущественными и создаст новый мир без них… они просто подложат под них заряд интеллектуального динамита и двинутся дальше. Именно поэтому они могут создавать такие вещи, как экстремальное программирование и Rails, которые встряхивают всю индустрию».

Делайте инвестиции – в них секрет успеха Agile.

В следующих разделах рассказывается об инвестициях, которые нужны вашим командам от вашей организации. Возможно, вы не сможете получить их все, поэтому я предлагаю вам альтернативы. При этом альтернативы возможны ценой снижения эффективности, поэтому лучше приложите все усилия, чтобы добиться как можно больше инвестиций. Я включил в список только те, которые важны.

Список необходимых инвестиций

Все команды Agile

• Получите поддержку руководства, команд и ключевых стейкхолдеров, как описано в главе 5.

• Создайте долговечные кросс-функциональные команды и все время назначайте людей только в эти команды (см. раздел «Выберите или создайте Agile-команды» текущей главы).

• Предоставьте каждой команде коуча, который поможет ей научиться быть эффективной слаженной командой (см. раздел «Выберите Agile-коучей» текущей главы).

• Поручайте работу командам, а не отдельным людям. Ожидайте от команд самостоятельного выбора подхода к ежедневному планированию и распределению задач (см. раздел «Делегируйте полномочия и ответственность команде» текущей главы).

• Направьте усилия руководителей командного уровня на управление общей системой работы вместо управления отдельными людьми и задачами (см. раздел «Измените стиль управления командой» текущей главы).

[11] Отрывок из статьи Мартина Фаулера *Enterprise Rails* (http://martinfowler.com/bliki/EnterpriseRails.html).

• Организуйте физические или виртуальные рабочие помещения для каждой команды (см. раздел «Организуйте рабочие помещения» текущей главы).

• Для получения первого опыта работы в Agile выберите командам значимую, но несрочную задачу (см. раздел «Выберите команде подходящую для обучения задачу» текущей главы).

• Замените водопадные политики управления на политики управления Agile (см. раздел «Смените водопадные подходы в управлении» текущей главы).

• Удалите, пересмотрите или научитесь обходить политики отдела кадров, препятствующие эффективной командной работе (см. раздел «Измените вредные кадровые политики» текущей главы).

Команды фокусировки

• Рассчитывайте на 1–4 месяца пониженной производительности каждой команды (см. раздел «Найдите время на обучение» текущей главы).

• Включите в команду людей, имеющих навыки в сфере деятельности пользователей и заказчиков (см. раздел «Выберите или создайте Agile-команды» текущей главы).

• Убедитесь, что в каждой команде есть тот, кто принимает решение, над чем команда будет работать, или команда имеет к нему свободный доступ (см. раздел «Выберите или создайте Agile-команды» текущей главы).

• Предоставьте каждой команде коуча, который сможет научить ее практикам фокусировки (см. раздел «Выберите Agile-коучей» текущей главы).

• Обеспечьте каждой команде доступ к стейкхолдерам или их представителям (см. раздел «Делегируйте полномочия и ответственность команде» текущей главы).

Команды поставки

• Рассчитывайте на 2–6 месяцев пониженной производительности каждой команды (см. раздел «Найдите время на обучение» текущей главы).

• Объедините в каждой команде все нужные навыки разработки, такие как тестирование и эксплуатация (см. раздел «Выберите или создайте Agile-команды» текущей главы).

• Предоставьте каждой команде коуча, который сможет научить ее практикам поставки (см. раздел «Выберите Agile-коучей» текущей главы).

• Доверьте каждой команде контроль над ее процессами разработки, сборки, тестирования и релиза (см. раздел «Делегируйте полномочия и ответственность команде» текущей главы).

• Для получения первого опыта работы в Agile выберите задачу, предполагающую написание кода с нуля (green-field codebase), если коуч не сочтет, что в этом нет необходимости (см. раздел «Выберите команде подходящую для обучения задачу» текущей главы).

• Разберитесь с вопросами безопасности, которые мешают коллективной разработке (см. раздел «Решите проблемы, связанные с безопасностью» текущей главы).

Команды оптимизации

• Рассчитывайте на 1–3 месяца пониженной производительности каждой команды (см. раздел «Найдите время на обучение» текущей главы).

• Включите в команду экспертов в области бизнеса, рынка и продукта (см. раздел «Выберите или создайте Agile-команды» текущей главы).

• Предоставьте каждой команде коуча, который сможет научить ее практикам оптимизации (см. раздел «Выберите Agile-коучей» текущей главы).

• Передайте каждой команде ответственность за ее бюджет, планы и результаты (см. раздел «Делегируйте полномочия и ответственность команде» текущей главы).

Найдите время на обучение

Изменения даются непросто, и нужно время, чтобы научиться чему-то новому. Освоение Agile поначалу замедлит работу ваших команд.

Насколько они замедлятся? Объективных критериев продуктивности в разработке программного обеспечения нет [Fowler2003], но, исходя из моего опыта, я бы предположил снижение на 10–20 % на первых порах. По мере освоения знаний и навыков Agile производительность команд будет расти. Она будет увеличиваться, пока команды не достигнут уверенного владения навыками, и тогда рост постепенно начнет выравниваться (рис. 4.1). Это называется *J-кривой*, и она характерна для всех значительных изменений. В главе 5 мы более подробно рассмотрим эти изменения.

Временны́е затраты обычно окупаются уже в первый год. Длительность изначального спада зависит от уровней свободного владения навыками, которые осваивает каждая команда, как объяснялось в предыдущей главе. Напомню:

• *фокусировка* – 1–4 месяца;
поставка – 2–6 месяцев;
• *оптимизация* – 1–3 месяца.

Рис. 4.1. Изменение производительности в Agile с течением времени

Эти периоды пересекаются, поэтому команда, обучающаяся навыкам *фокусировки* и *поставки* одновременно, будет менее продуктивна в течение 2–6 месяцев. Для сравнения: команда, которая сначала изучает *фокусировку* и позже переходит к обучению *поставке*, продемонстрирует два спада: 1–4 месяца – при обучении *фокусировке* и затем 2–6 месяцев – при обучении *поставке*.

Производительность команд Agile меняется и с других сторон. Команды Agile фокусируются на том, чтобы полностью завершить разработку одной функциональности, прежде чем переходить к следующей. Это особенно верно в отношении команд *поставки*, которые предпочитают создавать качественный продукт с самого начала, а не исправлять баги в конце. Это улучшает продуктивность и производительность, но (как ни странно) выглядит как замедление работы для людей, которые привыкли видеть одновременно несколько функциональностей в разработке.

В результате стейкхолдеры могут быть разочарованы темпом Agile-разработки, особенно в течение первого года, когда им приходится справляться сразу с тремя ударами: с реальными задержками из-за обучения команд, с ощущением задержки, вызванной необходимостью последовательно фокусироваться на завершении задач, и с затратами на завершение работ, которые велись до эксперимента с Agile и были объявлены «выполненными», *на самом деле* не являясь таковыми.

Это может приводить к тому, что команды будут отвлекаться от изучения Agile и фокусироваться только на поставке программного обеспечения, не закончив обучение. Такая ситуация контрпродуктивна для всех: команды будут измучены и расстроены, а организация потеряет свои инвестиции. Перед тем как команды начнут внедрять Agile, убедитесь, что руководители и все стейкхолдеры отдают себе отчет в том, что в первый год производительность снизится.

У вашей организации есть возможность купить время за деньги, наняв людей, которые помогут командам. Это не устранит полностью спад производительности, но сделает его менее значительным. Варианты такой помощи весьма разнообразны: периодическое наставничество, тренинги, помощь в разработке и имплементации процессов, каждодневный коучинг. Эффективнее всего нанять опытных специалистов-практиков, которые будут на постоянной основе тренировать каждую команду.

Решая, кого нанять, лучше игнорируйте бесчисленные схемы сертификации Agile. Многие из них – пустая трата денег. Большинство просто в течение нескольких дней переливают из пустого в порожнее, читая лекции «капитана Очевидность». Другие если даже и считаются отличными обучающими программами, то лишь благодаря конкретному преподавателю, а не самой сертификации. Поэтому оценивайте курсы независимо от сертификации, которую они рекламируют. Это относится и к найму консультантов и коучей. Попросите свою сеть контактов предоставить рекомендации, просмотрите общедоступные материалы, изучите отзывы.

Начав применять практики из этой книги, вы, вероятно, столкнетесь со специфическими проблемами и задачами. Найдите наставника, к которому сможете обратиться с вопросами. Эта услуга не обязательно должна быть платной: уважаемый коллега из компании, проходившей через подобное, местное сообщество пользователей, интернет-форум – все это будет подходящими вариантами.

Если нет времени на обучение…

Вы можете сделать спад производительности менее заметным за счет увеличения общих затрат, если начнете только с одного уровня *фокусировки* и постепенно переведете фокус на завершение задач.

Если ваша организация вообще не приемлет снижения производительности, значит, сейчас не лучшее время инвестировать в изменения. Если кажется, что подходящее время никогда не придет, – это предупреждающий сигнал. Вам нужно убедить руководство найти время для изменений, прежде чем продолжать.

Если нет средств на финансовую помощь…

С помощью этой книги, множества бесплатных онлайн-ресурсов и при наличии стремления к новым знаниям ваши команды могут самостоятельно научиться всему, что им нужно знать. Помощь извне, конечно, *полезна*, но не обязательна.

Отберите или создайте Agile-команды

Невозможно преувеличить важность команды в Agile-организации. Большинство организаций рассматривают людей как основной производственный ресурс. В Agile ресурсом являются *команды*.

Вашей организации нужно инвестировать в команды, которые будут:

• *кросс-функциональными* – члены команды в совокупности обладают всеми знаниями, которые нужны ей для достижения цели;

полностью вовлеченными – внешние специалисты могут время от времени приходить в команду, чтобы помочь, но основные члены команды должны посвящать все свое время только своей команде;

сплоченными – отношения между членами команды дружеские, конструктивные, позволяющие сотрудничать;

• *долгосрочными* – у команд могут уходить месяцы на то, чтобы понять, как наиболее эффективно работать вместе, поэтому сохраняйте состав команд как можно дольше.

Размер и состав каждой команды зависят от того, какие уровни свободного владения навыками вы выбрали. В разделе «Вся команда» главы 7 представлены все подробности. Краткое описание команд выглядит следующим образом.

• Команды *фокусировки* концентрируются на достижении бизнес-результатов. Им нужны люди, способные поставить себя на место пользователей и заказчиков, чтобы точно понять, что должно делать программное обеспечение. Если команда работает над задачей, ориентированной на пользователя, необходимы специалисты, имеющие навыки UI/UX (UI – User Interface – «пользовательский интерфейс», UX – User Experience – «опыт пользователя»). Команды также должны иметь способ определять, чем заниматься дальше. Лучше всего, если в команде есть люди с навыками и полномочиями, позволяющими делать это самостоятельно, однако члены команды могут работать и с кем-то извне.

• Команды *поставки* берут на себя ответственность за полный цикл поставки своего программного обеспечения. Им требуются все навыки, необходимые для сборки и развертывания программного продукта. Команда поставки должна брать на себя те виды ответственности, которые раньше передавались другим командам. Сюда входят управление сборкой, архитектура и администрирование данных, тестирование и эксплуатация.

• Команды *оптимизации* ответственны за успех своего продукта в бизнесе в широком смысле. Они также берут на себя ответственность за координацию со стейкхолдерами и принимают решения о продуктовых приоритетах. Им нужны эксперты в сфере бизнеса, рынка и продукта.

Возможно, у вас уже есть команды, которые отвечают всем требованиям. Если вы собираете новые Agile-команды, то можете выполнить шаги, описанные ниже. В любом случае вам понадобится вовлечь команду в процесс, как описано в разделе «Заинтересуйте команду» главы 5.

1. Определите цель каждой команды (см. раздел «Цель» главы 7).

2. Решите, сколько человек будет в каждой команде, основываясь на значимости цели команды и в зависимости от ограничений, описанных в разделе «Вся команда» главы 7.

3. Определите, какие навыки нужны в каждой команде.

4. Выберите людей с соответствующими навыками, которые наверняка смогут сотрудничать и хотят попробовать работу в Agile.

Если вы создаете или реорганизуете множество команд, то позвольте командам провести самостоятельный отбор. Этот способ удивительно эффективен в создании высокопродуктивных команд, которые будут в восторге от совместной работы. В книге *Creating Great Teams: How Self-Selection Lets People Excel* [Mamoli2015] рассказывается, как это работает.

Если вы не можете закрепить людей за определенной командой…

Успех Agile зависит от тесного сотрудничества, и он плохо работает, если нужные люди недоступны. Внешние ресурсы, приходящие время от времени, – это неплохо, но если вы не сможете заполучить людей, которые будут посвящать все свое время команде, есть вероятность, что Agile не будет работать.

Если члены команды не ладят друг с другом…

Для новой команды нормально проходить через трудный период, когда люди учатся работать друг с другом, поэтому не волнуйтесь, если первое время в команде происходит борьба. Коуч и менеджер команды могут помочь в урегулировании конфликтов. В разделе «Динамики команды» главы 11 эта тема разбирается более подробно.

Если вы не можете создать долгосрочную команду…

Разбивать отлично работающую команду – расточительно, но это не помешает вашим командам быть Agile.

Если вы не можете получить необходимых экспертов со знанием бизнеса, клиентов или пользователей…

Командам *оптимизации* необходим по меньшей мере один человек с навыками продакт-менеджера, но это не обязательно должен быть традиционный продакт-менеджер. Иногда разработчики, давно работающие в компании, знают продукт и рынок лучше, чем кто-либо еще. Если это ваш случай, то у вас есть все, чтобы приступить к работе.

Если ваши команды развивают навыки не на уровне *оптимизации*, то вам не нужен продакт-менеджер непосредственно в команде. Но вам все же понадобится некто с такой квалификацией, чтобы он работал в тесном контакте с командой, и вам нужны члены команды, которые могут представлять интересы клиентов и пользователей.

Вовлеченность бизнеса играет огромную роль в успехе команды. Это одно из тех свойств, которые отличают Agile от его предшественников. Приложите максимальные усилия, чтобы интересы бизнеса, клиентов и конечных пользователей были представлены в вашей команде. Если этого не сделать, то поставленное программное обеспечение, скорее всего, разочарует.

Если вы не можете получить необходимые вам навыки разработчиков…

Скорее всего, вы не сможете достичь навыков в *поставке*, но практики *поставки* вам все же стоит изучать и использовать в работе.

Выберите Agile-коучей

Каждой команде нужен коуч, который поможет ей научиться быть эффективной Agile-командой. В подразделе «Навыки коучинга» главы 7 содержатся подробности. Краткое описание коуча представлено ниже.

• Каждой команде необходим тот, кто может помочь ей научиться быть эффективной и сплоченной.

• Командам *фокусировки* нужен тот, кто может научить практикам планирования, описанным в части II.

• Командам *поставки* нужен тот, кто может научить техническим практикам, изложенным в части III.

• Командам *оптимизации* нужен тот, кто сможет научить практикам развития бизнеса, описанным в части IV.

Некоторые коучи могут охватить сразу несколько категорий. Каждый коуч может работать с одной или двумя командами.

Если вы не можете нанять на работу нужных вам коучей…

Вы можете воспитать собственных. Выберите старших специалистов-практиков, пользующихся уважением и доверием команды (если сразу не очевидно, кто они, то спросите членов команды, кого они могут рекомендовать), и предложите им попробовать себя в роли коучей. В этой книге есть все, что поможет им начать. Такие коучи, закрепленные за одной командой, – идеальный вариант.

Делегируйте полномочия и ответственность команде

Уважение к человеческим способностям лежит в центре философии Agile, и нигде это не очевидно настолько, как в подходе Agile к полномочиям и ответственности.

> Секрет первоклассного выполнения работы заключается в правильном понимании деталей, а никто не понимает их лучше, чем непосредственные исполнители этой работы… Имея необходимую квалификацию и направляемые лидером, они будут принимать наилучшие технические решения и воплощать их лучше, чем кто-либо другой смог бы сделать за них [Poppendieck2003].
> *Мэри и Том Поппендик*

С точки зрения организационных инвестиций это означает следующее.

• *Работа поручается командам, а не отдельным людям.* Команды сами решают, как разбить весь объем работы на задачи и кто в команде будет их выполнять. Возможно, вам понадобится изменить процесс распределения задач и другие рабочие процессы в соответствии с этим подходом. Это приводит к определенным последствиям в системе оценки эффективности, о чем мы подробнее поговорим в разделе «Измените вредные кадровые политики» текущей главы.

Команды сами разрабатывают свои рабочие процессы. В частности, команды должны иметь возможность свободно использовать простой, безынструментальный подход к планированию вместо того, чтобы привязываться к корпоративным инструментам. Руководство может ограничивать процессы команды, но причины каждого из этих ограничений должны быть четко сформулированы.

Команды фокусировки работают со стейкхолдерами, чтобы понимать потребности и приоритеты бизнеса. Организация должна обеспечить команде доступ к заинтересованным сторонам или их представителям.

Команды поставки контролируют процессы разработки, сборки, тестирования и релиза. Опять же, руководство может устанавливать ограничения на процессы команд, например, предписывать им следовать корпоративному конвейеру релизов, но нужно дать командам возможность разрабатывать и выпускать код в своем темпе, не дожидаясь других команд.

• *Команды оптимизации управляют своим бюджетом и планами продукта.* Менеджмент определяет цель каждой команды, общую стратегию и устанавливает бюджет. Кроме того, он осуществляет надзор в виде анализа бизнес-показателей. В рамках этой модели организация должна позволить командам самостоятельно решать, как достигать их целей и расходовать бюджет.

Если работу нужно поручить отдельным людям…

Если вашу организацию не устраивает то, что команды самостоятельно принимают решения по распределению своих задач, значит, в ней дефицит доверия, а доверие – требование Agile. Вы могли бы попытаться убедить людей изменить их мнение, попробовав командную работу с пилотной Agile-командой, но делайте это осторожно. Стиль управления «командуй и контролируй» обычно несовместим с Agile.

Если эта проблема не слишком широко распространена и дело лишь в нескольких менеджерах, которые не могут отпустить ситуацию, то прочитайте раздел «Измените стиль управления командой» текущей главы.

Если корпоративные инструменты не поддерживают командную работу…

Если в вашей компании используется система распределения задач, которую невозможно заменить, то одним из вариантов краткосрочного решения проблемы может стать создание для каждой команды «фантомного» человека, чтобы он принимал командные задачи. Другой вариант – члены команды могут рассматривать получаемые индивидуальные задания как командные.

В долгосрочной перспективе лучше все же исправить корпоративную программу.

Если команды должны использовать корпоративный инструмент отслеживания…

Одна из главных причин эффективности Agile-команд – способность улучшать и упорядочивать свои рабочие процессы. Корпоративные системы отслеживания, включая так называемые инструменты управления жизненным циклом Agile (Agile Lifecycle Management), ограничивают возможности команд. Как и множество продуктов, борющихся за место в вагоне локомотива Agile, эти инструменты имеют тенденцию настолько упускать смысл самого подхода, что начинают *снижать* гибкость команд.

Принуждение Agile-команд к использованию корпоративных инструментов отслеживания в их повседневной деятельности снижает их производительность. Если у вас нет выбора в этом вопросе, то можно применять две отдельные системы отслеживания: одну для легковесного подхода Agile и основную корпоративную систему. Более подробная информация доступна в подразделе «Корпоративные системы отслеживания» главы 10.

Если у команды нет доступа к стейкхолдерам…

В отличие от водопадных процессов, имеющих фазы предварительного сбора требований и бизнес-анализа, Agile-команды работают со стейкхолдерами в течение всего процесса разработки, уточняя планы и получая обратную связь. Не имея доступа к этим людям, команды не смогут сделать правильный продукт.

Если команда не может работать с одной или несколькими группами стейкхолдеров, то предоставьте ей доступ хотя бы к кому-то, кто представляет интересы этих групп. Выбирайте этого человека тщательно: качество разрабатываемого командой продукта будет напрямую зависеть от доступности этого человека и его способности корректно представлять потребности данной группы.

Если команда поставки не управляет своим процессом релиза…

Вы не сможете увидеть всех преимуществ владения навыками команд *поставки*, пока они не получат контроль над своим процессом релиза. Тем не менее практики *поставки* достаточно ценны, чтобы стремиться к мастерству в этой области. Вы сможете со временем избавиться от этой проблемы.

Если команда оптимизации не управляет своими планами создания продукта и расходами…

Командам *оптимизации* необходимо иметь возможность экспериментировать и адаптировать свои планы, а для этого им нужен контроль над планами и расходами. Без этого они не смогут достичь свободного владения навыками на уровне *оптимизации*.

Измените стиль управления командой

Когда команды сами определяют свои процессы, назначают себе задачи и координируют работу со стейкхолдерами, менеджеры командного уровня могут подумать, что им нет места в Agile. Но это далеко не так. Работа руководителя группы в Agile *различается*, но она не менее важна, чем в команде периода «до Agile». Больше информации об этом можно найти в разделе «Менеджмент» главы 10.

Поговорите с менеджерами об их новой роли и предложите им тренинг, если необходимо. Убедитесь, что *их* ожидания тоже соответственно изменились.

Если менеджеры не могут отпустить ситуацию…

Микроменеджмент раздражает, но в краткосрочной перспективе он не будет камнем преткновения. Однако он препятствует обучению, отбирая у команд возможность самостоятельно принимать решения. Микроменеджеры удлиняют сроки и повышают затраты, необходимые для достижения уровня свободного владения навыками[12].

Руководители часто занимаются микроменеджментом, когда не знают, чем еще им заниматься, или когда боятся, что им не найдется места в среде Agile. Заверьте их, что у них есть роль, показав, как она выглядит. Тренинг или хороший Agile-коуч могут в этом помочь.

[12] Спасибо Джорджу Динвидди за это замечание.

Организуйте рабочие помещения

Команды Agile активно сотрудничают и постоянно общаются друг с другом. Чтобы это общение было эффективным, потребуется помещение, приспособленное под потребности команды. Оно может быть как физическим, так и виртуальным. В разделе «Командная комната» главы 7 содержится более подробная информация.

Для команд, работающих в режиме личного общения, организация физического помещения может стать вашей самой дорогостоящей инвестицией. Вдобавок она является и наиболее ценной. Как обсуждается в разделе «Командная комната» главы 7, физические командные комнаты играют роль мультипликаторов производительности.

Однако, когда ваша команда только начинает работу, вы можете не знать, какого рода помещение ей нужно и даже приживется ли вообще Agile на долгий срок. Ваши команды, вероятно, тоже этого не знают. Команды-новички в Agile обычно недооценивают то, насколько им понравится совместная работа, и переоценивают свою тягу к приватности.

Так что это нормально – застраховать свои ставки на физическое рабочее пространство. Заложите на это средства в бюджете (в конце концов вам понадобится хорошее помещение для команды, если вы сработаетесь с Agile), но в краткосрочной перспективе вы можете реквизировать для каждой команды одну из больших комнат для собраний или часть открытого пространства офиса.

Что бы вы ни решили, начните работать над этим как можно раньше. Организация физического помещения занимает много времени.

Если команда удаленная…

Вы можете создать виртуальную командную комнату. В подразделе «Виртуальные командные помещения» главы 7 рассказывается, как это сделать.

Если вы не можете организовать физическое помещение для офисной команды…

Команды, работающие в режиме личного общения, также могут использовать виртуальные комнаты, но я очень не рекомендую это делать. В таком случае им придется столкнуться с худшими сторонами обоих вариантов: отсутствие гибкого подхода и регулярные поездки на работу, присущие офисной работе, в сочетании с трудностями общения, возникающими при удаленной работе.

Выберите команде подходящую для обучения задачу

У любой команды есть *цель*: ее место в общей стратегии организации. (См. раздел «Цель» главы 7.) Когда команда только начинает учиться Agile, важно выбрать цель, которая поможет в учебе. В практическом смысле это означает три вещи.

• *Цель, имеющая ценность, но не срочная.* Если команда будет сильно ограничена во времени, то ей будет трудно учиться. Члены команды по умолчанию вернутся к тому, что у них хорошо получалось в прошлом, чем будут тратить время на воплощение новых идей.

Автономная цель. Чем сильнее команда зависит от других, тем с бóльшими сложностями координации она, скорее всего, столкнется. Некоторых таких сложностей следует ожидать, но их переизбыток будет отвлекать команду от обучения.

• *Совершенно новая кодовая база* (green-field codebase). Команды, изучающие практики *поставки*, должны многому научиться, и это проще делать с нуля. В конце концов им придется

научиться работать с существующим кодом. Команды, у которых есть опытный коуч в области *поставки*, могут игнорировать это требование, если тот согласится. Таким же образом могут поступить и те команды, которые изучают уровни, отличные от *поставки.*

Если есть важный дедлайн…

Команде нужно много времени на обучение. Если сроки оставляют пространство для маневра, то все в порядке. Если нет, то обычно лучше отложить попытку внедрения Agile до момента, когда дедлайн закончится, или выбрать другие команды.

Если нет значимой работы с нуля…

Для команд важнее делать ценную работу, чем иметь кодовую базу с нуля. Однако, не имея опытного коуча, команды, только начинающие учиться практикам *поставки*, вероятно, будут испытывать сложности с уже существующим кодом. Будьте готовы к тому, что спад производительности продлится дольше, достижение навыков займет больше времени, а программисты команды будут более разочарованы.

Смените водопадные подходы в управлении

Руководство (Governance) – это процесс того, как работа согласовывается, отслеживается и управляется на высоком уровне. В большинстве организаций политики такого руководства предполагают водопадный подход в разработке. Часто это требует подготовки предварительной документации или четких переходов от фазы к фазе (phase gates). Все это означает предиктивный подход к планированию.

Для достижения лучших результатов нужно изменить политики руководства так, чтобы они стали соответствовать подходу Agile. Это значит удалить точки фазовых переходов и использовать адаптивный подход к планированию. Подробная информация доступна в подразделе «Agile-руководство» главы 10.

Если требуется водопадная модель управления…

Это расточительно и лишит команды части их гибкости, но при необходимости вы можете соблюдать водопадные политики руководства. Это возможно для нескольких пилотных команд, но будет лучше переключиться на Agile-руководство, прежде чем распространять Agile дальше.

Чаще всего руководство требует составления четких предварительных планов и бюджета. Простейший способ удовлетворить это требование – использовать сначала любой доступный вам подход, а затем, после того как вы пройдете через согласование проекта, начать Agile-часть процесса. В качестве альтернативы для команд, свободно владеющих навыками на уровне как *фокусировки*, так и *поставки*, вы можете заложить 4–8 недель на планирование, начать нормально работать и создать качественную дорожную карту (Roadmap). (См. раздел «Дорожные карты» главы 10.)

Как добиться успеха, используя водопадную модель

Agile подходит далеко не для каждой компании. И это нормально! Можно добиться успеха, используя водопадную модель. Если вы в компании, которой нужны предварительные планы или культура которой – «командуй и контролируй», то водопадная модель может быть более уместной.

Наиболее безопасно использовать *итеративный водопад*. Вместо того чтобы реализовывать один большой водопадный проект, что довольно

рискованно, начните серию небольших водопадных проектов. Каждый должен длиться не больше 3–6 месяцев и завершаться работающим программным обеспечением, действительно пригодным для выхода на рынок. Каждый проект должен содержать стандартные фазы водопада: анализ требований, архитектура и дизайн, реализация, тестирование – или тот вариант фаз, который предпочитает ваша компания.

Водопадная модель работает наилучшим образом в хорошо изученных предметных областях при наличии небольшой неопределенности. Постарайтесь взять на работу людей, весьма опытных в создании именно того типа программного обеспечения, который вы собираетесь реализовывать.

Другую предварительную документацию, такую как анализ требований или проектная документция, также можно подготовить с помощью действующих подходов до начала выполнения Agile-части вашей работы. Оставшуюся часть работы, которую необходимо выполнять для того, чтобы ваш рабочий процесс соответствовал водопадному подходу, можно распределить параллельно Agile-работе вместе с пользовательскими историями (см. раздел «Истории» главы 8), как и любые другие запросы.

Водопадный стиль управления несовместим со свободным владением навыками на уровне *оптимизации*, который основан на адаптивном планировании. Если от вас требуется соответствие водопадным политикам управления, ограничьтесь уровнями *фокусировки* и *поставки*.

Измените вредные HR-политики

Agile – это командный спорт, и несмотря на то, что на словах командная работа поддерживается, многие компании имеют политики, непреднамеренно сдерживающие ее. Любая политика, поощряющая людей конкурировать друг с другом, затруднит работу по Agile. Особенно деструктивный пример – таблицы ранжирования, где члены команды оцениваются методом сравнения друг с другом. Люди, оказывающиеся на верхушке таблицы, получают повышение, а тех, кто «на дне», увольняют независимо от их реальной производительности.

Похожий случай – руководители, которые ценят только материальный результат. В Agile-команде есть много способов внести свой вклад в успех. Примером может служить человек, который непосредственно не пишет код, но проводит много времени, воспроизводя баги, или человек, который работает «за сценой» над улучшением коммуникации.

Во многих организациях также развита *культура вины*, при которой на ошибки реагируют путем наказания виновных. В мировоззрении Agile, напротив, ошибки рассматриваются как возможность обучения. Например, не-Agile-организация может уволить программиста за непреднамеренное удаление критически важной оперативной базы данных. Agile-организация вместо этого спросит: «Какие методы сдержек и противовесов мы можем внедрить, чтобы предотвратить случайное удаление баз данных, и как мы можем упростить восстановление после подобных ошибок?»

Такие особенности корпоративной культуры часто отражены в HR-политиках (Human Resources, HR) в части продвижения и поощрения. Если карьера людей зависит от внешней причины, независимо от их реального влияния на производительность команды, то у ваших команд, вероятно, будут трудности в совместной работе по Agile.

Вам не удастся изменить культуру вашей организации за одну ночь, но вы можете начать работать над изменением HR-политик, а руководители могут изменить способ своего обращения с командами. Это занимает много времени, так что начните как можно раньше. Вам наверняка понадобится поддержка высшего руководства.

Часто лучше найти креативные способы применения существующих политик, чем отменить все политики сразу, что вдобавок может оказаться значительно сложнее. Кроме того, помните, что иногда руководство может применять какие-то практики по собственному усмотрению, поэтому если вы слышите, что что-либо «невозможно сделать», то это может быть из-за менеджера, которого вы уговариваете, а не из-за отдела кадров.

Если HR-политики не подлежат изменению…

Если изменить плохие HR-политики невозможно, то руководителям придется ограждать от них свои команды. Сделайте так, чтобы они и освоили Agile, и могли хорошо ориентироваться в корпоративной бюрократии.

Если у вас много команд, то ограничьте пилотный Agile командами, имеющими опытных руководителей. Используйте их опыт, чтобы запустить необходимые вам изменения политик.

Решите проблемы, связанные с безопасностью

Эти инвестиции обычно не проблема, но если они вдруг *становятся* проблемой, то могут затормозить вас намертво. Так что уделите им внимание, особенно если вы работаете в индустрии, имеющей повышенную чувствительность к безопасности.

Проблема в том, что работающие очно команды, которые используют такие практики, как парное программирование и групповое програмирование (моб-программирование) (см. соответствующие разделы главы 12), работают вместе за одним компьютером. Это может беспокоить с точки зрения безопасности, поскольку человек, который вошел в систему, не всегда тот же, кто сейчас сидит за клавиатурой. Фактически человек, который авторизовался в системе, может ненадолго отойти, чтобы поговорить с кем-то или посетить уборную. Набирая код, люди часто меняются (буквально каждые несколько минут), поэтому выходить из системы каждый раз, когда вводить код должен другой человек, и затем снова входить нереально.

Если ваши команды будут использовать эти практики, то привлеките людей из службы безопасности вашей компании и поработайте с ними над решением беспокоящих их вопросов. Обычно можно найти креативный способ поддерживать работу по Agile и одновременно соблюдать правила безопасности. Один общий подход – создать закрытую общую учетную запись для отдела разработки. Некоторые компании совмещают ее с отдельными рабочими станциями, выделенными специально отделу разработки, или виртуальными машинами на базе общих серверов. Использовать электронную почту и выполнять другие индивидуальные задачи можно на других ноутбуках, выданных индивидуально.

С этим связана проблема возможности отслеживания. Некоторые компании требуют, чтобы каждое внесенное изменение (commit) можно было отследить до его автора. Это требование можно выполнить, добавив инициалы или адрес электронной почты в коммит-комментарий команды. У Git есть соглашение о том, чтобы добавлять строку Co-authored-by в конце каждого коммит-сообщения[13].

Некоторые компании требуют, чтобы весь код перед релизом просмотрел еще один человек, помимо автора. В парном и групповом программировании это требование выполняется, но, скорее всего, вам понадобится модифицировать свой инструментарий, чтобы он позволил осуществить релиз кода, минуя дополнительную фазу проверки. Если полная отмена этого требования – не подходящий вариант, то вам, возможно, понадобится модифицировать инструменты так, чтобы пропускать проверку изменений, выполненных в соавторстве.

[13] Спасибо Джею Базузи за то, что обратил мое внимание на эти соглашения о commit-сообщениях (https://oreil.ly/7vSmz).

Если требования безопасности не допускают гибкости…

Вы можете потребовать, чтобы человек, авторизовавшись в системе, не отходил от компьютера. Если ему нужно отойти на минуту, то пусть он или переключает компьютер на другого пользователя, или вся работа останавливается, пока он не вернется. Это может вызвать гораздо больше разногласий, чем вы ожидаете, поэтому предпочтительнее использовать решения, позволяющие продолжать работу.

Кроме того, команды могут использовать инструменты, предназначенные для совместной удаленной работы, вместо того чтобы работать на одном компьютере. Это вызывает гораздо больше трений, чем все другие возможности, даже если члены команды сидят бок о бок, поэтому я не рекомендую это, если ваша команда уже не работает удаленно.

Если вам требуется дополнительный этап ревью кода…

Код, написанный в результате парного или группового программирования, уже может считаться прошедшим дополнительное ревью, поэтому команды могут начать не глядя штамповать отзывы об этом коде. Однако это тоже вызывает сложности, так что лучше убрать данное требование до того, как начинать широкое распространение Agile.

Руководство по устранению неполадок

Если Agile не работает для ваших команд и особенно если вы замечаете одни и те же проблемы в нескольких командах, то причина, вероятно, в недостаточных инвестициях. Ваши команды в большинстве случаев могут сами вам сказать, что им мешает, но если нет, то сверьтесь со списком часто встречающихся проблем, который приводится ниже.

Члены команды не пытаются применять новые практики

Команда не заинтересовалась Agile (см. «Заинтересуйте команду» главы 5); или

В команде нет коуча, который может научить ее практикам (см. раздел «Выберите Agile-коучей» текущей главы); или

На команду слишком давят, чтобы она давала результат (см. раздел «Выберите команде подходящую для обучения задачу» текущей главы).

Внутри команды слишком много конфликтов

Команды слишком часто переформируются (см. раздел «Выберите или создайте Agile-команды» текущей главы); или

Команда испытывает слишком большое давление (см. раздел «Выберите команде подходящую для обучения задачу» текущей главы); или

Менеджеру команды приходится постоянно помогать улаживать конфликты (см. раздел «Измените стиль управления командой» текущей главы); или

Политики отдела кадров поощряют конкуренцию (см. раздел «Измените вредные кадровые политики» текущей главы).

Члены команды не сотрудничают друг с другом

Команда не заинтересовалась Agile (см. раздел «Заинтересуйте команду» главы 5); или

Члены команды не имеют возможности посвятить все свое время команде или не ладят друг с другом (см. раздел «Выберите или создайте Agile-команды» текущей главы); или

В команде нет коуча, который может научить ее сотрудничеству (см. раздел «Выберите Agile-коучей» текущей главы); или

Распределение или отслеживание задач требует индивидуальной работы (см. раздел «Делегируйте полномочия и ответственность команде» текущей главы); или

Рабочее пространство не подходит команде (см. раздел «Организуйте рабочие помещения» текущей главы); или

Команда испытывает слишком большое давление (см. раздел «Выберите команде подходящую для обучения задачу» текущей главы); или

Менеджер команды раздает индивидуальные задания (см. раздел «Измените стиль управления командой» текущей главы); или

HR-политики поощряют индивидуальную работу (см. раздел «Измените вредные кадровые политики» текущей главы).

Команда тратит слишком много времени на предварительную оценку, планирование и отслеживание работ

Команда вынуждена использовать корпоративную систему отслеживания задач (см. раздел «Делегируйте полномочия и ответственность команде» текущей главы); или

От команды требуют составления предварительных планов и детальных прогнозов (см. раздел «Смените водопадные подходы в управлении» текущей главы); или

Команде нужно развивать свободное владение навыками на уровне *фокусировки* (см. часть II).

Программное обеспечение, созданное командой, не делает то, что нужно стейкхолдерам

У команды нет доступа к нужному представителю бизнеса, или ей нужны люди с большим опытом в сфере деятельности заказчика и пользователей (см. раздел «Выберите или создайте Agile-команды» текущей главы); или

В команде нет коуча, который может научить ее работе со стейкхолдерами (см. раздел «Выберите Agile-коучей» текущей главы); или

У команды нет доступа к стейкхолдерам (см. раздел «Делегируйте полномочия и ответственность команде» текущей главы).

Команде тяжело привлекать внимание стейкхолдеров

Стейкхолдеры не поддержали инициативу с Agile (см. раздел «Заручитесь поддержкой стейкхолдеров» главы 5); или

У команды отсутствуют необходимые навыки в сфере деятельности заказчика (см. раздел «Выберите или создайте Agile-команды» текущей главы); или

Стейкхолдеры не считают работу команды актуальной или значимой (см. раздел «Выберите команде подходящую для обучения задачу» текущей главы).

Программное обеспечение, созданное командой, делает то, что хотят заказчики и пользователи, но не имеет коммерческого успеха

У команды нет доступа к нужному представителю бизнеса или опыта в бизнесе (см. раздел «Выберите или создайте Agile-команды» текущей главы); или

Команде нужна более подходящая цель (см. раздел «Выберите команде подходящую для обучения задачу» текущей главы); или

Команде нужно развивать свободное владение навыками на уровне *оптимизации* (см. часть IV); или

Команда свободно владеет навыками в *оптимизации*, но не может контролировать свои планы и расходы (см. раздел «Делегируйте полномочия и ответственность команде» текущей главы).

Программное обеспечение, разрабатываемое командой, имеет очень долгие циклы выпуска версий, много багов или эксплуатационные проблемы

В команде нет людей, уверенно владеющих навыками на уровне *поставки* (см. раздел «Выберите или создайте Agile-команды» текущей главы); или

В команде нет коуча, который может научить ее практикам *поставки* (см. раздел «Выберите Agile-коучей» текущей главы); или

Команде нужно развивать свободное владение навыками на уровне *поставки* (см. часть III); или

Команда свободно владеет навыками на уровне *поставки*, но не может полностью контролировать свою разработку, выпуск версий и эксплуатацию (см. раздел «Делегируйте полномочия и ответственность команде» текущей главы); или

Команда учится работать с существующим кодом (см. раздел «Выберите команде подходящую для обучения задачу» текущей главы).

Разработка идет медленнее, чем ожидалось

Команда завершает предыдущие задачи, которые были до Agile, или все еще учится (см. раздел «Найдите время на обучение» текущей главы); или

Команде нужно больше коучинга (см. раздел «Выберите Agile-коучей» текущей главы); или

Рабочее пространство не подходит командам (см. раздел «Организуйте рабочие помещения» текущей главы); или

Команде нужно развивать свободное владение навыками на уровне *поставки* (см. часть III); или

Команда не может полностью контролировать свой процесс разработки (см. раздел «Делегируйте полномочия и ответственность команде» текущей главы); или

Команда работает с существующим кодом (см. раздел «Выберите команде подходящую для обучения задачу» текущей главы); или

Команда ограничена требованиями руководства, отданными на высоком уровне (см. раздел «Смените водопадные подходы в управлении» текущей главы); или

Команда ограничена требованиями безопасности (см. раздел «Решите проблемы, связанные с безопасностью» текущей главы).

Глава 5. Инвестируйте в изменения

Вы решили, что Agile сделает вашу команду более успешной. Вы знаете, какие уровни предлагают наилучший компромисс между затратами и выгодой. Вы определились с инвестициями, которые должна сделать ваша компания. Теперь осталось понять, как все это сделать на практике.

Осознание изменений

Изменения разрушительны, и внедрение Agile – не исключение. *Степень* разрушительности зависит от того, скольких команд они коснулись и насколько успешно вы управляете этими изменениями. Если у вас одна команда, жаждущая попробовать Agile при полной поддержке со стороны организации, то это не должно быть большой проблемой. Если вы пытаетесь изменить 50 команд в организации, не знакомой с идеями Agile… что ж, это *очень* большая проблема.

Понять, как люди реагируют на изменения, можно с помощью модели изменений Вирджинии Сатир (рис. 5.1)[14]. Согласно рисунку, существует пять ступеней перемен. Они применимы к Agile следующим образом.

1. Имеющийся статус-кво. Это стиль работы, который был до Agile. Он удобен и всем хорошо знаком. Все знают, чего от них ждут и как делать свою работу. Некоторые, тем не менее, не очень довольны и считают, что Agile должен помочь. Они хотят перемен.

2. Сопротивление. Люди, желающие перемен, начинают получать поддержку, и какие-то изменения в направлении Agile становятся возможными. Это называется *внешним элементом изменения*. Люди начинают по-разному реагировать на возможность перемен. Многие противостоят им. Они говорят, что в Agile нет необходимости, что вряд ли его внедрение будет успешным или что это пустая трата времени. Некоторые злятся. Чем больше людей касаются изменения, тем больше сопротивления вы встречаете.

3. Хаос. Проект изменений одобрен, и команды начинают использовать практики Agile. Старые методы работы и привычные ожидания больше не действуют. Люди растеряны, находятся в замешательстве, и настроения в коллективе переменчивы. В какие-то дни все хорошо, в другие – все плохо. Люди периодически демонстрируют ребяческое поведение. Производительность и моральный климат ухудшаются.

4. Интеграция. С течением временем, практикуясь, люди начинают осваивать новые методы работы. Они открывают для себя тот аспект Agile (его называют *трансформирующей идеей*), который особенно убедителен для них (для каждого человека он свой). Люди начинают осознавать возможности, которые дает им Agile, и прикладывают реальные усилия, чтобы Agile заработал. Ощущение хаоса снижается, моральный климат улучшается, производительность растет.

5. Новый статус-кво. Коллектив прошел через изменения. Новые методы работы по Agile уже комфортны и знакомы, и люди достаточно уверены в себе, чтобы продолжать добавлять небольшие изменения. Производительность стабилизировалась на более высоком уровне, чем была до перемен, и продолжает постепенно расти по мере того, как люди экспериментируют с внедрением небольших новшеств.

[14] У Стивена Смита есть хорошая статья об этой модели (https://oreil.ly/1KQ38), включающая советы о том, как помочь команде на каждом этапе.

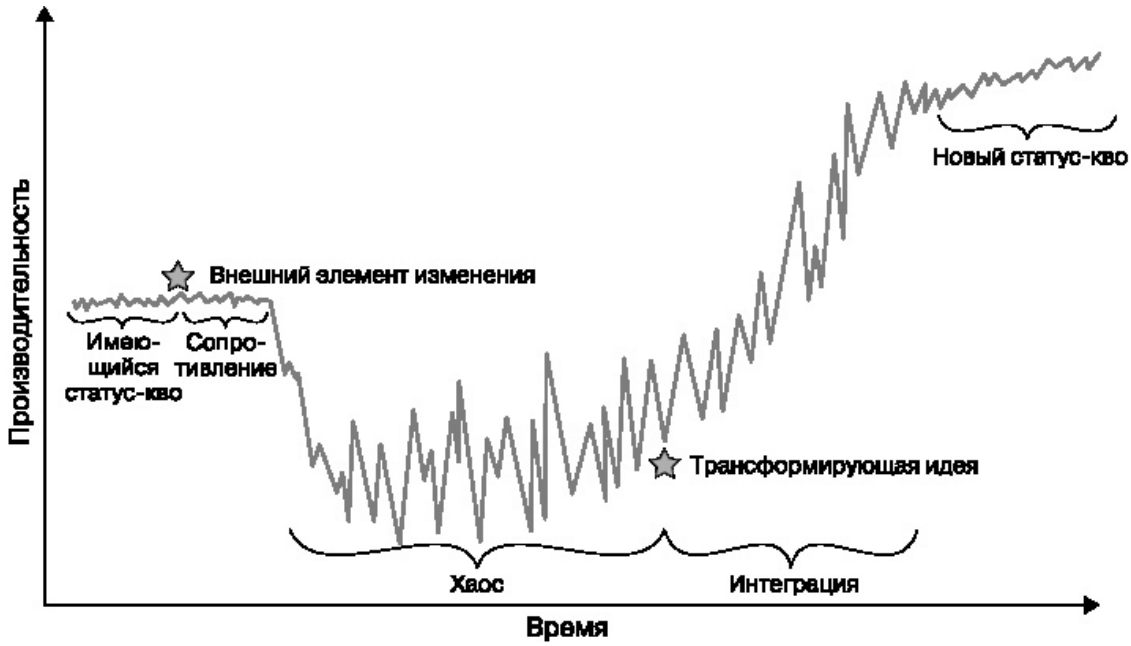


Рис. 5.1. Модель изменений Вирджинии Сатир (The Satir Change Model)

Такие реакции на изменения неизбежны. Попытки ускорить процесс только ухудшают ситуацию. Вот почему организациям следует отвести время на обучение (см. раздел «Найдите время на обучение» главы 4). Обратите внимание на параллели между моделью Сатир, представленной на рис. 5.1, и *J*-кривой, показанной на рис. 4.1 (см. выше).

Каждый участник проходит через эти ступени в своем темпе. Длительность периода изменений и глубина хаоса зависят от того, в какой степени затронута повседневная жизнь человека. Участник, действующий на периферии процесса, реагирует легче, чем тот, кто непосредственно является частью новой Agile-команды. Индивидуальные особенности личности также имеют значение; одним людям нравится пробовать все новое, в то время как другие хотят стабильности и предсказуемости.

Вы можете снизить (но не полностью устранить!) хаос, применив технику, которой я научился у Дианы Ларсен: поддержка, информация и структура (Support, Information, Structure – SIS)[15].

• *Поддержка.* Помогите людям разобраться, как выполнять работу в изменившейся окружающей среде. Организуйте обучение, коучинг или найдите любые другие способы оказать помощь людям, не внушая им мысль, что их оценивают. Сделайте необходимые инвестиции, описанные в главе 4. Убедитесь, что у каждого есть кто-то, с кем он может поговорить на работе или в частной жизни, когда чувствует себя перегруженным.

Информация. Обеспечьте прозрачность информации в отношении того, что происходит, что известно и с чем еще предстоит определиться. Не оставляйте без внимания озабоченность людей карьерными вопросами. Если вы можете сделать это честно, то дайте твердое обещание, что никто не будет уволен в результате изменений. Сообщайте даже больше, чем считаете нужным[16].

Структура. Людям нужна твердая почва под ногами, поэтому предоставьте им дорожную карту грядущих изменений. Если вы используете эту книгу в качестве основы для ваших изменений, то раздайте всем ее копии и расскажите, какие части в данный момент используете. Когда что-то неясно, объясните, что нужно сделать, чтобы внести определенность, и когда,

[15] Спасибо Диане Ларсен за помощь в составлении этого списка.

[16] Диана говорит: «Коммуницируйте, пока вас не затошнит. И даже потом продолжайте».

по вашим ожиданиям, это произойдет. Если нужен промежуточный шаг, например временные команды, то дайте ясно понять, что это временно, и объясните, что будет дальше.

Масштабные изменения

Изменения, затрагивающие множество людей, гораздо более разрушительны, чем те, которые касаются лишь нескольких команд. Это разрушение имеет тенденцию усиливаться. Появляются слухи, люди начинают беспокоиться о своих рабочих местах, и грядущие перемены становятся предметом ежедневного обсуждения.

Большие перемены (непосредственно затрагивающие 30–70 и более человек) требуют профессионального управления изменениями. В зависимости от размеров вашей организации ваш отдел кадров может иметь в своем составе специалистов по изменениям, которые могут вам помочь. Если вы нанимаете внешних консультантов Agile на этот проект, то поинтересуйтесь их опытом и подходом в области управления изменениями.

Руководители организаций часто недооценивают важность управления изменениями. Это большая ошибка! Используя терминологию модели Сатир, к тому времени, как вся остальная организация узнает об изменениях, руководство организации уже восприняло «трансформирующую идею», которая избавила их от ощущений неприятия и хаоса. Для руководства изменения уже кажутся очевидными и необходимыми. Так почему кто-то может быть не согласен?

И тогда они запускают процесс изменений и сталкиваются с огромным сопротивлением и противодействием со стороны людей, которые только начали проходить сквозь *собственные* ступени неприятия и хаоса. И это способно загубить на корню весь проект изменений.

Правильный процесс управления изменениями не может предотвратить все разрушения, но *может* их минимизировать. Не скупитесь. Если вы не готовы потратиться на экспертную помощь, то ограничьте масштабы вашего проекта изменений в направлении Agile всего несколькими командами за раз.

Процессы изменений

Кайдзен – общий термин в сообществе Agile. Это японское слово, означающее «совершенствование». В Agile-сообществе этот термин означает *непрерывное постепенное* совершенствование[17].

Непрерывное совершенствование – неотъемлемая часть Agile, так не нужен ли *кайдзен* в первую очередь для того, чтобы направить ваши усилия к Agile? Парадоксально… но, наверное, нет. *Кайдзен* предназначен для улучшения существующих методов работы. Если в вашей компании особое значение имеют документы, то *кайдзен* поможет вам рационализировать документооборот. Если культура вашей организации основана на обвинениях, то *кайдзен* поможет вам точнее устанавливать вину. Но он не поможет вам перейти с любой культуры на Agile.

Чтобы перейти с одного стиля работы на другой, вам понадобится другое японское слово – *кайкаку*. Оно означает «трансформационное изменение». Вместо того чтобы постепенно улучшать существующие процессы работы, как это делает *кайдзен, кайкаку* фундаментально меняет основополагающий подход.

---

[17] Термин «кайдзен» был импортирован в Agile из концепции бережливого производства (lean manufacturing), которая, в свою очередь, основывается на революционной производственной системе компании Toyota, – отсюда и японский термин. Рифмуется с английским I win – «я выигрываю».

Все лучшие команды Agile, которые я знаю, начинали с *кайкаку*. Они разобрались, что им нужно от Agile, как они собираются инвестировать в достижение этого результата, и пошли ва-банк.

Я вижу команды Agile, которые работают посредственно, значительно чаще, чем те, которые функционируют прекрасно. Первых объединяет то, что их компании в свое время не пошли ва-банк. Они пытались проложить свой путь к Agile через *кайдзен.* Поначалу кажется, что это работает, но потом все неизбежно останавливается. Люди выгорают из-за несовпадения идей Agile и ценностей компании. Они устают впитывать новые идеи. Изменения вызывают переутомление, прогресс начинает тормозиться и после многолетних усилий полностью останавливается. По иронии судьбы, разрушения при таком способе продвижения к Agile значительно более длительные, чем при *кайкаку*.

Если ваша компания – новичок в идеях Agile (даже если вы уже используете это слово), то вам нужен *кайкаку*. Определите зоны применения, выделите необходимые средства, и пусть каждая команда начнет использовать все соответствующие практики сразу. Возможно, это звучит пугающе, однако на самом деле это быстрее и безопаснее, чем осваивать практики постепенно.

Если у вас много команд, то, вероятно, безопаснее действовать постепенно, но даже тогда *кайкаку* – лучший вариант для вас. Вместо того чтобы в соответствии с *кайдзен* постепенно внедрять Agile во многих командах, примените *кайкаку*, чтобы полностью внедрить Agile в *подгруппе* команд. Если нужно двигаться еще более мелкими шагами, то начните всего с одной команды и, возможно, только с одного уровня *фокусировки*. Затем добавьте уровень *поставки*. Далее добавьте еще одну команду, возможно, вместе с уровнями *фокусировки* и *поставки*. По мере накопления опыта увеличьте размер шага изменений.

Команды, уже работающие по Agile, могут использовать *кайдзен* для достижения лучших результатов в рамках их нынешних областей компетентности. Более подробная информация доступна в главе 11. Чтобы добавить новые области (например, если команда компетентна на уровне *фокусировки* и хочет стать таковой на уровнях *поставки* и *оптимизации*), лучше всего выбрать *кайкаку*. Новые уровни требуют новых инвестиций и серьезных изменений, и лучше делать все сразу.

Успешный *кайкаку* требует дисциплины и заботы. Рассмотрим, с чего все начинается.

Заручитесь поддержкой руководства

Agile требует поддержки руководства. Без нее несоответствие между Agile-практиками вашей команды и никак не связанной с Agile культурой вашей организации будет вызывать постоянные трения. Если вы сами руководитель, постарайтесь сделать так, чтобы *ваш* непосредственный руководитель был на вашей стороне, и будет совсем хорошо, если коллеги тоже вас поддержат.

1. Начните с разговора

Все начинается с разговора. Часто проще всего разговаривать один на один, вы получите наилучший результат, если будете общаться лично или, по крайней мере, по видеосвязи. Начните с одного из влиятельных руководителей, которому вы доверяете, и завербуйте его в свои союзники. Он поможет разобраться, с кем еще вам нужно поговорить и как найти к ним подход.

В этих разговорах, начиная с того самого первого руководителя, говорите о проблемах в разработке ПО, с которыми сталкивается ваша организация. Возьмите за основу преимущества, описанные в предисловиях к частям II–IV, и расскажите о том, насколько, по вашему мнению, разработка программного обеспечения могла бы вестись лучше в вашей компании. Не

монополизируйте разговор; вовлеките в него вашу аудиторию. Вкратце обрисуйте преимущества свободного владения навыками на каждом соответствующем уровне и поинтересуйтесь, какие уровни важны по мнению собеседников. Спросите почему. Слушайте больше, чем говорите. Прежде всего фокусируйтесь больше на том, *что они могут получить* и *к каким потерям может привести бездействие*, чем на продвижении Agile ради него самого. На самом деле, учитывая степень всеобщего недопонимания, что такое Agile, вам может быть лучше вообще не упоминать этот термин.

2. Получите одобрение экономичного покупателя

Ваша конечная цель – поговорить с человеком, обладающим полномочиями распоряжаться инвестициями, которые понадобятся вашей команде. В продажах этот человек называется экономичным покупателем (economic buyer).

Экономичные покупатели часто бывают окружены своего рода привратниками, работа которых – беречь время экономичного покупателя. Они предложат вам передать всю информацию им, чтобы *они* могли презентовать ее покупателю. Они не пытаются украсть вашу идею; они пытаются помочь экономичному покупателю сберечь его время. Иногда они пытаются убедить вас, что покупатель на самом деле – это они, даже если у них в действительности нет необходимых полномочий.

Не дайте себя обмануть. Заручиться поддержкой привратников полезно и часто даже необходимо, однако этого недостаточно. Они не смогут одобрить нужные вам инвестиции. Вам нужно разговаривать с реальным экономичным покупателем.

До этого разговора фокусируйте ваше общение на выгодах Agile: что на кону. Вопрос инвестиций, скорее всего, будет отвлекать или даже вызывать беспокойство, поскольку люди, с которыми вы будете разговаривать, не имеют полномочий управлять инвестициями.

Когда разговор с экономичным покупателем наконец состоится, вашей целью будет получить его принципиальное согласие инвестировать в Agile. Вероятнее всего, ваше время будет ограниченным, так что удерживайте фокус на общей картине. Кроме того, часто лучше провести эту встречу в формате диалога, а не презентации со слайдами, хотя в вашей конкретной ситуации, скорее всего, ваши союзники смогут предложить вам наилучший подход.

Во время встречи с экономичными покупателями говорите о том, чего они хотят от их организации и как Agile может в этом помочь. Это может сработать еще лучше, если кто-то из руководителей, кому они доверяют, будет говорить от вашего имени.

Заполучив в команду союзников экономичного покупателя, начните говорить об инвестициях. Не перегружайте собеседников излишними подробностями; кратко перечислите несколько ключевых инвестиций, необходимых каждому уровню (врезка «Список необходимых инвестиций» в главе 4 поможет вам подготовиться), а также то, как эти уровни соотносятся с пожеланиями покупателя. Спросите его, какой компромисс между размером инвестиций и выгодой он считает приемлемым. Если вы предложите несколько вариантов, а не потребуете ответа «да или нет», это понизит шансы категорического отказа.

При условии, что экономичный покупатель в принципе согласился инвестировать в Agile или по крайней мере считает возможным дальнейшее рассмотрение этого вопроса, попросите разрешения подготовить конкретное предложение. Спросите, что он хотел бы увидеть в нем в целом, чтобы его одобрить. Попросите рекомендовать конкретное контактное лицо, спонсора, с которым вы будете работать, сообщите дату, когда предложение будет готово (лучше сделать его в течение одного-двух дней, поэтому хорошо иметь готовый черновик документа), и спросите, когда ожидать ответа.

В конце разговора попросите разрешения напомнить. Вероятнее всего, с вами не свяжутся в оговоренный день, поэтому хорошо иметь возможность сказать что-то вроде: «Напоминаю, как вы и просили».

3. Сделайте официальное предложение

Если вы добрались до этого этапа, то поздравляю! Вы преодолели самое важное препятствие. Сейчас вам нужно пройти через официальное предложение.

Степень формальности предложения зависит от вашей организации. Ваш спонсор и ваши союзники помогут вам разобраться, как лучше оформить предложение, помогут доработать его, правильно преподнести экономичному покупателю. Действуйте оперативно, вежливо и *настойчиво*.

В вашем предложении опишите преимущества, которые может получить ваша организация, и инвестиции, которые ей необходимо сделать. Будьте конкретны. В главе 3 описываются преимущества в целом, а в предисловиях к частям II–IV дана более подробная информация. Перенесите эти преимущества на вашу конкретную ситуацию, на инвестиции, которые экономичный покупатель готов сделать, и объясните, что в реальности это означает для вашей организации.

При подготовке инвестиционного аспекта вашего предложения прочитайте главу 4 и переведите каждый шаг в конкретный запрос. В каких-то инвестициях вам, вероятно, придется пойти на компромисс. В главе объясняется, как это сделать. Но постарайтесь избежать слишком больших компромиссов. В конечном счете именно инвестиции делают возможными преимущества Agile.

Выход на экономичного покупателя

Приведу пример того, как вовлеченность руководства работает в сложной ситуации; он взят из моего собственного опыта в роли консультанта. Это не было обычным проектом по переходу организации к Agile, поэтому у вас все будет иначе, но этапы обсуждения в организации наверняка будут похожими.

Изначально ко мне обратился менеджер по проектированию компании, насчитывающей несколько сотен инженеров и порядка 45 команд разработчиков программного обеспечения. Этот руководитель искал кого-нибудь, кто возглавил бы маленькую команду. Мы поговорили, и я узнал, что у них были сложности с взаимодействием команд. Я предложил свою помощь в решении этой проблемы.

Этот руководитель нашел мои идеи касательно масштабирования довольно содержательными и представил меня своему вице-президенту по проектированию. Мы поговорили спустя две недели. Ему понравилось то, что я сказал, и еще через полторы недели мы обедали с его боссом – директором по продукту. Он и был экономичным покупателем.

Во время того обеда директор по продукту высказал ряд опасений, касающихся его организации. Мы обсудили несколько способов, как я могу помочь, и в качестве продолжения запланировали встречу между мной и их директором по управлению продуктами, который также присутствовал на том обеде.

Я встретился с директором через неделю. Он не являлся покупателем, поэтому моей целью было не убедить его, а узнать больше информации о компании. Я также хотел убедиться, что мы правильно понимаем друг друга, поскольку нам предстояло тесно сотрудничать. К счастью, мы были на одной

волне, и мой собеседник запланировал еще одну встречу с участием нас двоих и директора по продукту.

Эта встреча состоялась на следующей неделе. Наша предыдущая встреча за обедом была нашим шансом познакомиться друг с другом, тогда как *эта* встреча была моим шансом заручиться поддержкой директора по продукту.

Я не пытался развернуть рекламную кампанию. Я никогда не считал полезным *говорить* людям, что они должны хотеть. Вместо этого я задавал вопросы. У меня была реальная цель: я хотел понять потребности директора по продукту и проинформировать его о тех моментах, которые чреваты трудностями или даже проблемами. Я спросил его, как он представляет себе успех, как он поймет, что успех достигнут, и как, по его ожиданиям, в итоге это повлияет на бизнес.

К концу встречи, которая заняла час, я узнал достаточно, чтобы обрисовать примерный подход и ценовой диапазон. Я спросил, правильно ли я все понял, и мой собеседник ответил утвердительно. Мы ударили по рукам, и я обещал к концу следующего рабочего дня представить детальное предложение, что и сделал. Его рассмотрели в течение недели (директор держал меня в курсе того, что происходило в компании) и одобрили на следующей неделе, внеся небольшие поправки.

Окончательное одобрение заняло еще некоторое время, которое ушло на то, чтобы пробиться сквозь корпоративную бюрократию, однако на тот момент это был уже решенный вопрос. От первой встречи до первого одобрения прошло пять встреч и семь недель. Это довольно быстрый результат для организации такого размера, но они были очень мотивированы. У них была серьезная проблема, которая блокировала прогресс, и они верили, что я могу помочь им решить ее.

Вот те ингредиенты, которые понадобятся и вам: мотивирующая проблема и вера в то, что вы можете ее решить.

Если это выглядит слишком трудозатратным…

Этот осторожный процесс привлечения сторонников предназначен для случаев, когда поддержка под вопросом: если вы работаете со множеством команд, запрашиваете большие инвестиции или работаете в бюрократической организации, для которой идеи Agile не комфортны (даже если люди активно используют это *название*).

Но иногда все это не так сложно. Иногда вы просто помогаете одной маленькой команде стать более Agile. Если вы и ваш руководитель уже имеете все полномочия, чтобы выделить необходимые вам инвестиции, то просто сделайте это!

Если руководство считает, что они уже Agile…

Некоторые организации (а в наши дни таких *много*) думают, что они уже Agile. Одна организация говорила мне: «Мы пост-Agile!» Или вы можете услышать: «Мы agile с маленькой буквы “a”, а не Agile с большой буквы “A”!» Но сравнив философию, описанную в главе 1, с тем, как действует организация, вы поймете, что это далеко не так.

Спорить о терминологии бесполезно. Если организация хочет сказать, что она Agile, или пост-Agile, или экстра-супер-пупер-Agile, то пусть говорит. Вместо того чтобы спорить, фокусируйтесь на текущей ситуации: проблемах, с которыми сталкиваются ваши команды, преиму-

ществах, которые организация может получить, и на том, какие инвестиции потребуются для этого.

Если руководство не поддерживает…

Если поначалу вы не можете найти понимание и поддержку у руководства, то не отчаивайтесь. Попробуйте поставить себя на их место. Как Agile может помочь им получить то, что им нужно? Если ответ «не может», то, вероятно, он им и не нужен. Выберите другой подход к разработке программного обеспечения, такой, который лучше впишется в культуру вашей организации. Один из вариантов можно найти во врезке «Как добиться успеха, используя водопадную модель» в главе 4.

Если у вас есть руководитель, которому вы доверяете, то обратитесь к нему за помощью и советом. Если его нет, то попробуйте провести информационное интервью: поговорите с руководителем из другой компании, проходившей через это недавно. (Они могут попытаться нанять вас на работу. Взаимовыгодный вариант, win/win!) Во врезке «Смените организацию» далее в этой главе рассматривается ряд идей на этот случай.

На заре развития Agile, когда это было еще стихийным общественным движением, множество команд приняли экстремальное программирование самостоятельно, при слабой поддержке или согласии руководства, а то и вообще без них. Вы тоже *могли бы* попробовать это сделать. Но я не рекомендую. Опыт команд, которые попытались это сделать, показал, что кому-то (чаще всего руководителю проекта) в конце концов приходилось все время преодолевать разрыв между корпоративной культурой и философией Agile. Это неблагодарная работа, которая привела к выгоранию команды.

Некоторые используют метод Kanban, чтобы мотивировать организационные изменения[18]. Kanban представляет существующие методы работы таким образом, чтобы подчеркнуть узкие места в исполняемой работе и показать стоимость задержек. Его достаточно легко внедрить, и он может мотивировать переход организации к Agile.

Kanban является *кайдзен*-подходом к изменениям, он медленный и работает только до определенного предела, но очень эффективен и может привести к появлению разрешения на *кайкаку*. Более подробную информацию вы можете найти в книге [Anderson2010].

Если ничего из того, что вы делаете, не меняет ситуацию, то задумайтесь, а что нужно *вам*. Представьте, что статус-кво не изменится, поскольку этого, вероятно, и не случится. И тогда или вас это устраивает, или (а часто это как раз такой случай) пришло время перейти в другую, более подходящую вам компанию.

Изменить организацию

Вы можете изменить свою организацию или сменить ее.

Мартин Фаулер

Изменить организацию изнутри непросто, но возможно. Это отнимает много сил и времени, и не всегда оказывается, что оно того стоило. Поэтому более простой способ изменить организацию – сменив работу – может оказаться более разумным выбором. Для тех, кто все же хотел бы попытаться, ниже приведены 13 советов, почерпнутых из моего собственного опыта изменения организации изнутри.

1. Задумывайтесь о своих мотивах. Отвечает ли Agile в наивысшей степени интересам организации или он нужен вам по личным причинам? Есть ли у вас время, энергия и энтузиазм, чтобы проповедовать эти изменения?

[18] Обратите внимание, что метод Kanban – это значительно более широкое понятие, чем доска Kanban, применяемая некоторыми командами для визуализации планирования.

Есть ли у вас стратегия поиска выхода из проблем, которые могут возникнуть вследствие ваших усилий? Если ответ на любой из этих вопросов – «нет», то смена работы может оказаться лучшим выходом.

2. Организуйте надежную сеть поддержки. Изменения в направлении снизу вверх – неблагодарная работа, часто ведущая к разочарованию. Полагайтесь на семью и друзей, уходите домой вовремя и не живите рабочими проблемами в нерабочее время.

3. Находите для себя маленькие удовольствия. Без поддержки сверху организационные изменения в значительной мере находятся вне вашего контроля. Находите небольшие ежедневные рабочие задачи, благодаря выполнению которых вы будете чувствовать удовлетворение.

4. Не сдавайтесь. Небольшие изменения накапливаются. Поначалу эффект от ваших усилий будет незаметным, но они будут постепенно менять стиль мышления людей в сфере решения проблем. В конце концов, по достижении некоего порога все внезапно начнет меняться.

5. Уважение – ваша валюта. Чем больше люди уважают вас, тем больше ваш авторитет. Завоевывайте авторитет своими действиями и уважительно относитесь к другим, даже в мыслях.

6. Оставайтесь внутри сферы своего влияния. Изменения снизу требуют постоянного повторения. Пытайтесь вносить изменения только в тех отделах организации, с которыми вы находитесь в постоянном контакте.

7. Пополняйте ряды сторонников. Найдите хотя бы одного человека, который уважает вас и ваши идеи и имеет большее влияние, чем вы. Привлеките его к распространению ваших идей.

8. Находите пробелы. Люди должны хотеть изменений, но они будут их хотеть, только если это дает им что-то, что они не могут получить другим способом, или если это убережет их от потери чего-то ценного для них. Фокусируйтесь на этих моментах.

9. Вникайте в причины. Всегда есть причины, почему что-то делается именно так, а не иначе, и вы можете сделать свое вмешательство более эффективным, если разберетесь, что это за причины.

10. *Повторяйте.* Продвигайте свою идею изменений снова и снова, разными способами и разным людям. Но старайтесь не быть назойливым.

11. *Не критикуйте все подряд.* Выберите что-то одно и работайте с этим. Если вы будете находить проблемы во всем, то люди просто перестанут вас слушать.

12. *Не ищите признания.* Если вы успешны, то люди начнут повторять ваши идеи, как если бы они были их собственными. Это не плагиат. Ваши усилия на самом деле могут изменить мышление людей, при этом они даже не будут это осознавать. Пусть они так думают. Они будут более усердно трудиться ради собственных идей.

13. *Будьте осторожны в своих желаниях.* Если ваш проект изменений будет успешен, то готовы ли вы взять на себя ответственность за то, что будет дальше?

Если вы захотите прочесть историю изменений, вдохновившую автора на эти советы, то можете найти ее онлайн на веб-странице [Shore2006]. Подробное руководство по изменениям изнутри содержится в книге *More Fearless Change: Strategies for Making Your Ideas Happen* [Manns2015].

Заинтересуйте команду

Agile ставит интересы людей на первое место, поэтому для вас не должно стать сюрпризом, что нужно получить согласие вашей будущей Agile-команды на то, чтобы попробовать новый подход. *Можно* заставить людей согласно кивнуть, стиснув зубы, но (и я говорю это, основываясь на моем нелегком опыте) это обычно приводит к потере кадров.

Когда меня просят помочь компаниям с внедрением Agile, я всегда разговариваю с каждой командой отдельно от ее руководителей. Нужно дать членам команды возможность высказаться в комфортных для них условиях, не опасаясь возмездия. Пригласите на встречу и коуча команды, и если вы сами руководитель, то начните разговор с того, что поддержите решение команды, каким бы оно ни было. Затем дайте людям поговорить с коучем без вас.

Вы или коуч можете сказать команде, что ее выбрали возможным кандидатом на тестирование Agile. Я обычно объясняю, почему руководство заинтересовано в Agile, какие выгоды он принесет организации и как это повлияет на членов команды лично. Я также объясняю, что изменение рабочих привычек – это стресс и команде следует ожидать хаоса (обычно на период до трех месяцев), пока каждый разберется, как сработаться с Agile. Еще я часто рисую набросок и поясняю модель изменений Вирджинии Сатир (см. рис. 5.1).

Я говорю: «Если вы согласитесь, то я предложу вам попробовать стандартный вариант Agile в течение трех месяцев. После этого мы оценим, что работает, что нет, и попробуем что-то улучшить[19]. Спустя шесть месяцев у вас будет возможность принять решение – продолжаем мы работать с Agile или остаемся с тем, что есть сейчас».

Затем я даю всем возможность задать вопросы. У команд обычно множество вопросов касательно процесса, но один вопрос возникает всегда: «Что будет, если мы скажем “нет”?» Мой ответ всегда один и тот же: «Ничего не произойдет». Это важно! *Должна быть* реальная возможность наложить вето. Если вы не дадите людям возможность сказать «нет», то их «да» ничего не будет значить. Давая людям шанс отказаться сейчас и конкретную возможность передумать позже, вы позволяете им в безопасном режиме попробовать что-то новое.

Выделите достаточно времени на то, чтобы ответить на все вопросы. Встреча обычно занимает около часа, но иногда может затянуться. Ответив на все вопросы, напомните, что не будет никаких последствий для проголосовавших против. Еще раз подчеркните это. И затем приступите к голосованию.

Если команда настроена скептически…

Скептический настрой – нормальное явление, его следует ожидать. Будьте честны с вашим коллективом: изменения разрушительны, но результаты себя оправдывают. Поясните практики, которые, по вашему мнению, люди поначалу могут счесть странными или даже раздражающими, например парное программирование. Это поможет снизить градус скептицизма и в будущем облегчит процесс внедрения этих практик.

Может быть полезно подчеркнуть, что это *эксперимент* и за командой будет последнее слово в вопросе, придерживаться Agile или нет.

Если несколько членов команды против…

Если не согласны лишь несколько человек, то попросите их объяснить причины и подумайте, можно ли разрешить эти разногласия. Если не получается, то предложите им пока воз-

[19] Это не совсем правда. Agile подразумевает постоянное совершенствование, поэтому мы оценим, что работает, а что нет, уже в течение нескольких недель. Но по истечении трех месяцев мы делаем паузу, чтобы дать более значительную оценку.

держаться от суждений и присоединиться к остальным членам коллектива на ближайшие шесть месяцев.

Если они все равно не согласны, то предложите им перейти в другую команду, при условии, что руководство одобрит это решение. Если же это невозможно или вы не знаете, кто именно не согласен (в случае анонимного голосования), что ж, значит, эта команда – неподходящий кандидат.

Если большинство членов команды против…

Если команда не согласна, то вам придется выбрать другую. В моей практике это случалось редко, но *так бывает*. В одном из случаев это было потому, что коллектив не верил, что их организация даст им достаточно времени на обучение. Теперь, оглядываясь назад, я понимаю, что они были правы, и хорошо, что мы тогда не приступили к изменениям.

Если люди обманывают насчет своего согласия…

Иногда люди голосуют за Agile, при этом внутренне выступая против. С этим вы ничего не можете сделать, кроме как приложить все усилия, чтобы люди не чувствовали себя принуждаемыми. Пытаться угадывать чужие мысли непродуктивно.

Даже если никто не обманывает сейчас, природа изменений такова, что время от времени всех будут посещать сомнения. Вам придется работать с этими сомнениями по мере их появления. И будет полезно иметь возможность напомнить людям, что они согласились принять участие в эксперименте длиной в шесть месяцев, что есть точная дата его окончания и если все не заработает к этой дате, то они смогут отказаться. Отнеситесь к ним с сочувствием и уважением; изменение привычек требует времени, и люди могут чувствовать, что устои, на которые они привыкли опираться, полностью утрачены.

Конец ознакомительного фрагмента.