Кен Швабер
Джефф Сазерленд

Софт за 30 дней

Как Scrum делает невозможное возможным

Джефф Сазерленд

Софт за 30 дней. Как Scrum делает невозможное возможным

«Манн, Иванов и Фербер»

2012

УДК 658.81:004.4
ББК 65.290с51-32

Сазерленд Д.

Софт за 30 дней. Как Scrum делает невозможное возможным / Д. Сазерленд — «Манн, Иванов и Фербер», 2012

Прочитав эту книгу, вы познакомитесь с методикой Scrum и узнаете, как этот нестандартный подход работает и как начать применять его в своем бизнесе на примере процесса разработки программного обеспечения. Гибкие технологии Agile и Scrum позволят вам осуществить то, что раньше казалось абсолютно невозможным, – создать полноценный работающий программный продукт всего за 30 дней. Эта книга поможет руководителям и менеджерам компаний, которые хотят покончить с дорогим и медленным циклом разработки ПО. На русском языке публикуется впервые.

УДК 658.81:004.4
ББК 65.290с51-32

Содержание

Кен Швабер и Джефф Сазерленд Софт за 30 дней. Как Scrum делает невозможное возможным

Информация от издательства

Издано с разрешения John Willey & Sons International Rights, Inc., и литературного агентства Александра Корженевского

Благодарим за помощь в подготовке издания компанию ScrumTrek в лице Алексея Пименова

* * *

Посвящается Икуджиро Нонаке, Бабатунде А. Огунайке и Хиротака Такеучи с благодарностью за их наставления и вдохновение

Введение

Мы, Джефф и Кен, работаем в индустрии программного обеспечения в общей сложности 70 лет.

Мы были разработчиками программного обеспечения, менеджерами в IT-организациях и компаниях программного обеспечения и владельцами компаний по девелопменту и сервисному обслуживанию софта. Более 20 лет назад мы создали процесс, который позволил сотням организаций делать программный продукт лучше. Наша работа распространилась шире, чем мы когда-либо могли себе представить, ее плодами воспользовались миллионы людей, и мы в восторге от результатов, которых они смогли добиться.

Это не первая наша книга о создании софта, однако это первая книга, которую мы написали для тех, кто сам не разрабатывает программное обеспечение, – скорее она адресована лидерам организаций, выживание и конкурентное преимущество которых зависят от программного обеспечения. Эта книга для лидеров, которые могут получить выгоду от быстрой поэтапной разработки программного обеспечения и при этом добиться максимально возможного возврата инвестиций. Эта книга для лидеров, которые сталкиваются с деловыми и технологическими сложностями, затрудняющими разработку софта. С ее помощью такие лидеры могут помочь своим организациям достичь этих целей, приумножить внутренние возможности, увеличить количество предложений продуктов и многое другое.

Это книга для руководителей компаний, исполнительных директоров и старших менеджеров, которым необходимо, чтобы их организация производила лучшее программное обеспечение за более короткий срок, с меньшими затратами, большей предсказуемостью и низкими рисками.

Для этой аудитории у нас есть послание.

Возможно, вы имели негативный опыт разработки программного обеспечения в прошлом, но индустрия уже перешла на новый уровень. Сфера программного обеспечения радикально улучшила свои методы и результаты. Неопределенность, риск и потери, к которым вы привыкли, перестали быть неизбежностью. Мы работали со множеством организаций в сфере программного обеспечения, которые уже перешли на новый уровень. Хотим помочь это сделать и вам.

Мы покажем, как повысить стоимость бизнеса, используя процесс, поставляющий законченные фрагменты функционала разрабатываемого программного обеспечения как минимум каждые 30 дней.

Кроме того, мы расскажем, как определить приоритеты по созданию функционала и получить их «а-ля карт», то есть как пожелаете.

Инструментарий в этой книге поможет вам увеличить скорость разработки, доведя ее до современных стандартов, и в итоге получить желаемый результат.

Это софт за 30 дней.

Раздел I. Почему любой бизнес в мире может создать программное обеспечение за 30 дней

Мы обращаемся к каждому лидеру в организации, который хочет создавать лучшие программные продукты с лучшими характеристиками и предсказуемостью. Индустрия программного обеспечения изменяется и радикально улучшается. Неопределенность, риск и потери, к которым вы привыкли, перестали быть неизбежными. У нас за плечами 20 лет работы с организациями, которые уже перешли на новый уровень. Мы хотим, чтобы и вы сделали этот шаг и были способны создавать полезное, качественное программное обеспечение с управляемыми рисками.

Мы обращаемся к вам по двум причинам. Во-первых, в течение 40 лет вы страдали от плохого обслуживания в индустрии программного обеспечения, не намеренно, но неизбежно. Мы хотим вернуть ваше доверие. Во-вторых, программное обеспечение – это уже не только специализированный инструментарий для профессионалов. Программы теперь в нашем обществе выполняют все более и более важные операции. Мы хотим, чтобы вы были способны создать программное обеспечение, на которое мы все можем надежно полагаться.

Мы надеемся, что сможем достичь этих целей с помощью нашей книги.

Вопреки всему не сдавайтесь. Вы не должны больше мириться с ужасным программным обеспечением прошлого. Двигайтесь дальше. В этой части книги мы разберем, почему до сих пор разработка программного обеспечения была столь плохой. Далее мы покажем, как оно улучшилось и какие два явления этому способствовали. Затем мы продемонстрируем, как применять наш подход, чтобы добиться успеха.

1. Кризис в программном обеспечении: неправильный процесс разработки приводит к неправильным результатам

Ваша организация, будь то коммерческая, некоммерческая или государственная компания, должна быть в состоянии выдавать полезный продукт путем создания, настройки и использования программного обеспечения. Без программ ваша способность достичь целей как бизнес-лидера сильно ограничена, если не полностью невозможна. Но, несмотря на эту потребность, разработка программного обеспечения – исторически ненадежный, дорогостоящий и подверженный ошибкам процесс[1]. Это создает проблему: вам нужно программное обеспечение, но вы не можете получить его в нужное время, по приемлемой цене и с уровнем качества, которое делает его использование удобным.

Действительно, The Standish Group в своем CHAOS Report 2011 года обнаружила, что половина проектов по разработке программного обеспечения с 2002 по 2010 год описывались либо как трудные для выполнения, либо как полный провал. Только 37 % были классифицированы как успешные (рис. 1.1). The Standish Group скромно называет успешным проект, который обеспечивает требуемый функционал в срок и по заявленной цене.

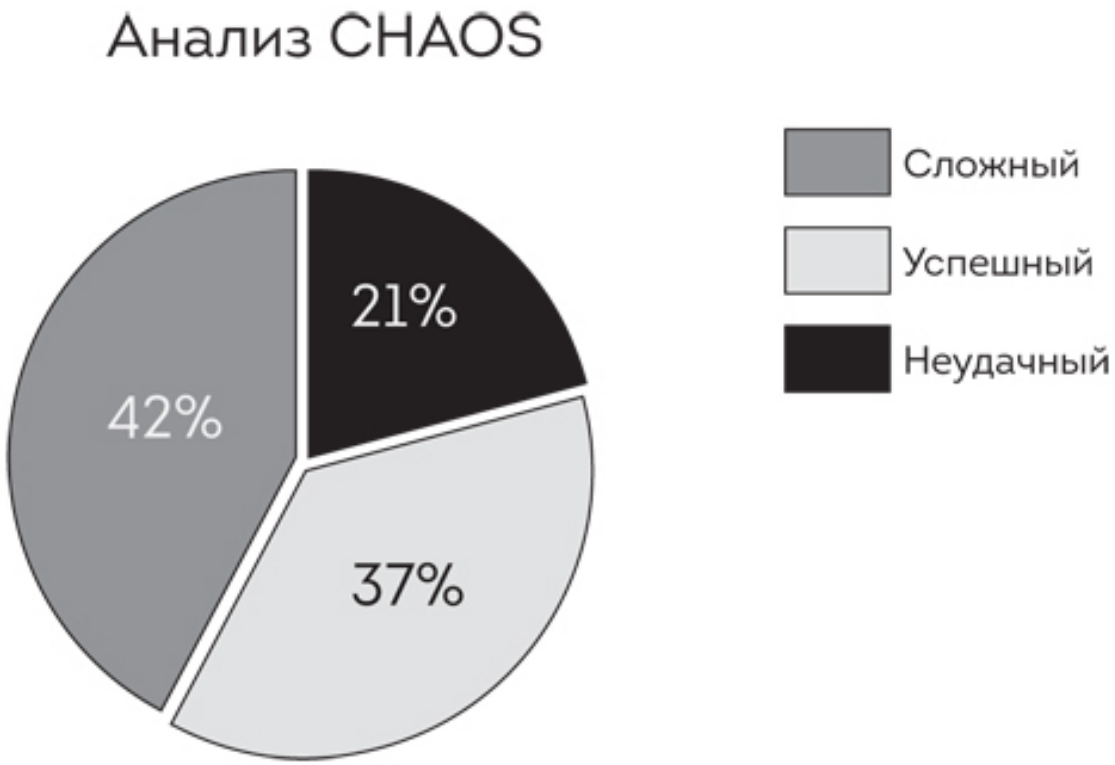

Рис. 1.1. Риски традиционного метода разработки программного обеспечения

[1] Апрель 11, 2005. Отчет Forrester Corporate Software Development Fails to Satisfy on Speed or Quality («Корпоративная разработка программного обеспечения не может удовлетворить скорости и качества»). Корпоративные разработчики продолжают разочаровывать: к осени 2004 года опрос Forrester 692 влиятельных участников индустрии, которые заказывают разработку программного обеспечения, показал, что около трети недовольны временем, затрачиваемым на разработку, и такое же количество недовольно качеством предоставляемых приложений. Одна пятая опрошенных недовольна обеими позициями.

Способность приспосабливаться к изменениям, управлять рисками или изначальная ценность программного обеспечения не рассматривалась.

Вероятность того, что проект разработки программного обеспечения будет успешным, невелика. Если вы пытаетесь сделать что-то важное, включающее в себя разработку программного обеспечения, то, вероятно, должны быть обеспокоены. Индустрия программного обеспечения, будучи медленной, дорогой и непредсказуемой, подведет вас. Если бы софт не был столь важным, вы бы, скорее всего, совсем не стали бы инвестировать в него.

И вы не одиноки. К примеру, проект «Страж» Федерального бюро расследований (ФБР) недавно столкнулся с проблемами, и его полностью обновили, используя идеи и процессы, описанные в этой книге.

Информация, касающаяся проекта «Страж», взята из общедоступных отчетов Министерства юстиции США. До того как вы определите это как частный случай, являющийся особенностью работы правительства, подумайте: если крупное министерство смогло кардинально улучить свой метод разработки программного обеспечения, то сможет и ваша организация.

Пример: проект «страж» федерального бюро расследований

Все записи, которые были созданы или получены в рамках того или иного расследования ФБР, собраны в папки. В 2003 году ФБР решило оцифровать дела и автоматизировать связанные процессы, чтобы агенты могли быстро сравнивать дела и обнаруживать связи между ними. Проект назвали «Страж».

В марте 2006 года ФБР приступило к разработке «Стража», целью проекта было создание базы для более чем 30 тысяч конечных пользователей, агентов ФБР, аналитиков и административного персонала. По предварительной оценке, на разработку и запуск «Стража», что должно было произойти к декабрю 2009 года, выделили 451 миллион долларов. Согласно первоначальному плану ФБР, разработка «Стража» должна была пройти в четыре стадии. Проекта поручили корпорации Lockheed Martin, практикующей традиционный процесс разработки программного обеспечения.

К августу 2010-го ФБР потратило 405 из 451 миллиона долларов бюджета «Стража», при этом предоставив функционал только двух из четырех стадий. Хотя эти результаты и улучшили систему управления расследованиями ФБР, они не обеспечивали всех полезных функций, которые предполагались первоначально.

В связи с превышением стоимости и сроков разработки в июле 2010 года ФБР выпустило распоряжение о прекращении сотрудничества и приостановке оставшихся двух стадий проекта.

До этого момента в ФБР использовали традиционный метод девелопмента и теперь решили опробовать новый подход с целью достичь лучших результатов. Мы разработали этот новый подход, названный Scrum, в начале 90-х. Тот самый CHAOS Report The Standish Group, который классифицировал только 37 % проектов успешными, продемонстрировал как различаются результаты традиционного подхода к разработке и тех, которые используют быстрый Scrum-подход (рис. 1.2).

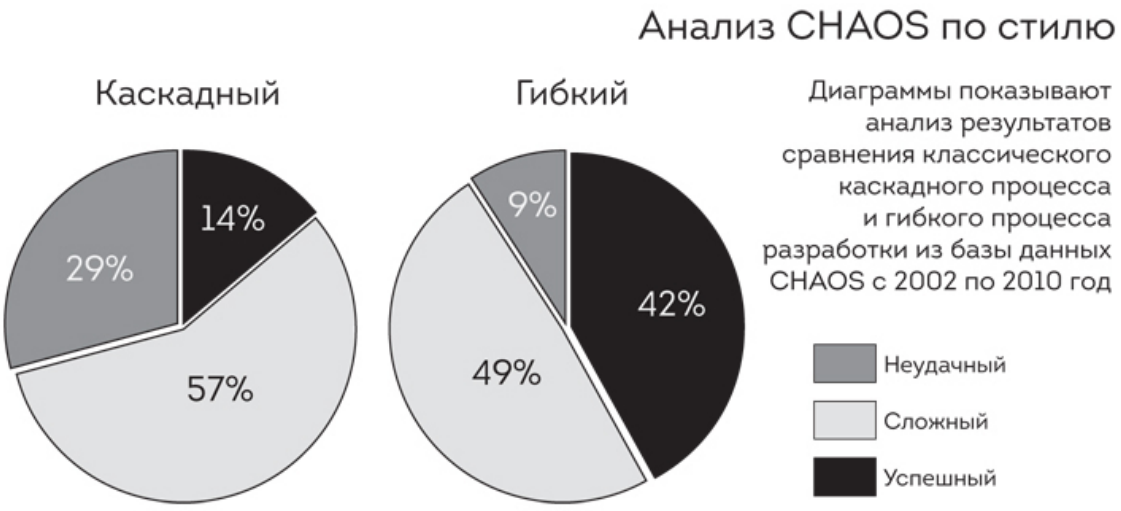

Рис. 1.2. Гибкие проекты в три раза более успешны

В частности, отчет показывает, что 42 % проектов, использующих быстрый метод, добиваются успеха. Что касается традиционных проектов, то здесь успешны только 14 %. Мы полагаем, что в дополнение к стандартным определениям успеха The Standish Group Scrum-проекты также обеспечивают более быстрое реагирование на изменяющиеся потребности клиентов, позволяют снижать риски и в конечном счете предоставляют более высокое качество программного обеспечения.

К 2009 году в ФБР были приняты на работу новый директор по информационным технологиям (CIO) и новый технический директор (CTO) с опытом управления организациями, создающими программное обеспечение на основе нашего метода. Они решили проверить, сможет ли этот более быстрый подход к разработке помочь ФБР. В 2010-м CTO сообщил департаменту правосудия, что намерен изменить подход к разработке проекта «Страж». Он утверждал, что новый подход оптимизирует процессы принятия решений и позволит ФБР остаться в рамках предусмотренного бюджета. ФБР сообщило генеральному инспектору Министерства юстиции, что сможет закончить разработку «Стража» в пределах оставшегося бюджета в течение 12 месяцев после возобновления проекта. Проведенный перед этим компанией Mitre аудит проекта показал, что для завершения «Стража» традиционным методом потребуются шесть лет и дополнительные 35 миллионов долларов.

ФБР перевело всю команду по разработке проекта «Страж» в подвал здания в Вашингтоне и сократило персонал с 400 человек до 45, 15 из которых были программисты. Технический директор ФБР лично возглавил проект. Целью управления разработкой стало предоставление части функционала «Стража» каждые 30 дней. Все приращения функционала обязаны были соответствовать окончательным характеристикам, это не должен был быть «сырой продукт». Каждые три месяца ФБР объединяло три ранее полученные части в единый пилотный проект.

К ноябрю 2011-го, через год после возобновления работы с использованием нового подхода, все стадии проекта «Страж» были закончены. Программное обеспечение установили для пилотной группы агентов ФБР, оснащение оставшихся агентов новым программным обеспечением было запланировано к июню 2012 года. ФБР удалось закончить проект «Страж» за 30 миллионов долларов в течение года. Экономия средств составила более 90 %. Сотрудники так же усердно трудились и над первыми стадиями проекта «Страж», но сам подход к разработке программного обеспечения отбрасывал их назад. После применения нового метода, описанного в этой книге, они продолжили работать столь же усердно, как и

раньше, но наградой им стали гораздо лучшие результаты. Если такая организация, как ФБР, смогла сделать это, почему не может ваша?

Неправильный подход: предиктивные процессы

Процесс, который ФБР изначально использовало для «Стража», мы называем предиктивным, или последовательным. По сути, до 2005 года большинство проектов по разработке программного обеспечения использовали предиктивные процессы, которые при определенных обстоятельствах наиболее подходят и могут обеспечить успех. Эти обстоятельства, однако, скорее исключение, чем правило. Если кто-то может создать окончательное видение, определить все требования к нему и затем разработать детальный план по воплощению их в жизнь, тогда предиктивный процесс будет эффективным. Но любое отклонение от изначального видения, требований или плана создает большой риск. При частой смене требований бизнеса и технологий, которые присутствуют в реальных условиях, очень редко встречается, чтобы все элементы проекта оставались неизменными. Как результат, о чем и сообщает отчет The Standish Group, 86 % проектов по разработке программного обеспечения, основанных на предиктивных процесах, неуспешны. На самом деле мы считаем использование предиктивных процессов самой распространенной причиной проблем при разработке программного обеспечения.

Организации, с которыми мы сотрудничаем, стараются изо всех сил, чтобы увеличить процент успеха своих проектов по разработке программного обеспечения. Они ищут помощи у нас из страха, что их организация процесса девелопмента ПО выходит из-под контроля. Существующий процесс подвел их, а альтернативных подходов они не знают. Проблемы с разработкой программного обеспечения приносят их организациям огромные потери, и они вынуждены мириться с этим, поскольку должны создавать программное обеспечение на конкурентном уровне.

Руководители и менеджеры обычно описывают проблемы, с которыми сталкиваются, следующим образом.

1. Выпуск продукта занимает все больше и больше времени. «Каждый релиз занимает больше времени, требует больше усилий и затрат до момента передачи потребителю. Несколько лет назад выпуск новой версии занимал 18 месяцев, теперь на разработку, комплектацию и установку требуется 24 месяца. И при этом каждый релиз становится стрессом и требует значительных усилий. Мы тратим все больше, при этом получаем все меньше и меньше.

2. Срыв графиков релизов. «В проспектах мы даем обязательства нашим нынешним либо потенциальным клиентам, которые делают определенные приготовления в соответствии с нашим графиком релиза. Им нужен наш релиз с обещанным нами функционалом именно тогда, когда запланировано. И мы обычно подводим их в самый последний момент. Их планы нарушаются, они теряют деньги и доверие в глазах своих клиентов. Следовательно, мы можем не получить от них новые заказы, их отзывы о нашей работе вряд ли будут хорошими, и они могут начать поиск другой компании разработчиков».

3. Создание стабильной версии перед релизом занимает все больше и больше времени. «Мы установили разработчикам четкие, неизменяемые даты выполнения заказа. Они уложились в эти сроки, но программное обеспечение было нестабильным, не функционировало, как требуется. У нас даже не получилось отгрузить этот софт как бета-релиз, так что мы смогли получить отзывы только от ограниченного количества пользователей. Дефекты оказались настолько значительными, что наши бета-тестеры отказались участвовать. Нам

потребовалось еще девять месяцев, чтобы выпустить релиз, и даже тогда он был нестабилен и требовал множества доработок и оправданий перед пользователями».

4. Планирование занимает слишком много времени и выполняется неправильно. «Мы выяснили, что разработка и развертывание релизов занимают слишком много времени, потому что мы не планировали достаточно хорошо в начале работы. Наши требования были не четко определены и не полностью разработаны, а оценки включали больше догадок, чем возможно. Чтобы это исправить, теперь мы больше времени тратим на планирование. Новые идеи продолжают поступать постоянно. Так как люди пересматривают планы, они находят части, которые должны быть переделаны или уточнены. Теперь мы тратим гораздо больше времени на планирование, чем раньше, но наш график плывет, и периоды стабилизации программного обеспечения все еще большие и ужасные. Несмотря на наши значительные усилия, во время разработки происходят изменения, которые не были и не могли быть учтены во время планирования».

5. Изменения трудно вносить в процессе разработки. «Текущий процесс не приспособлен к внесению изменений. Мы тратим много времени на планирование всего в начале разработки, и все необходимые этапы предусмотрены планом. Но часто необходимо вносить критические изменения или новые функции близко к дате релиза. Для этого мы должны настроить всю ранее проделанную работу для приспособления нового изменения. Это очень сложно, потому что трудно предсказать, как именно это изменение повлияет на стабильность программы. Даже если оно очень важное, есть ощущение, что для его приспособления в процессе разработки требуется в сто раз больше времени, чем если бы знать об этом изменении в самом начале. Но что мы можем сделать? Если важное изменение не внести в текущую разработку или релиз, нужно ждать два и более года до следующего релиза».

6. Ухудшение качества. «Мы знаем, что не должны давить на разработчиков, чтобы получать все, что запланировано, и с учетом изменений вовремя, но наш бизнес страдает от проблем планирования, срыва сроков и изменений. Мы требуем от разработчиков ужесточения работы для соблюдения сроков. Каждый раз после таких требований они сокращают срок выпуска за счет ухудшения качества программного обеспечения либо сокращения времени тестирования на стабильность. Результат настолько плохой, что мы возвращаемся на стадию стабилизации или выпускаем некачественный продукт».

7. Авралы наносят моральный вред. «Мы обращаемся с людьми так, как нам бы не хотелось. Однако у нас есть обязательства, и бизнес требует, чтобы все, кто работает над проектом, выходили в выходные и задерживались по вечерам. Их семьи и здоровье страдают. Как следствие, у нас есть проблемы с наймом хороших разработчиков, а наши лучшие разработчики уходят в другие организации. Персонал настолько деморализован, что его производительность ухудшается, несмотря на увеличение часов работы».

Этих примеров достаточно, чтобы обескуражить любого руководителя, занимающегося разработкой программного обеспечения. Несмотря на 20 лет титанических усилий и огромных затрат, в этой отрасли к началу 90-х был достигнут незначительный прогресс в обеспечении успешного результата проектов по разработке программ. Процесс, который мы опишем в этой книге, устраняет подобные проблемы.

Неправильные результаты: провал проекта

Использование традиционного, или предиктивного, процесса разработки программного обеспечения – основная причина, лежащая в основе провала многих проектов. Предиктивный процесс, также называемый каскадным, зависит от точности плана проекта и неизменности исполнения. Он полагается на перечисленные ниже факторы.

1. Требования не меняются и полностью понятны. Любые изменения в требованиях нарушают план. Требующиеся изменения в плане создают значительные изменения в уже проделанной работе, часто превращая ее в полностью бесполезную. К несчастью, более 35 % всех требований меняются в процессе типового проекта по разработке программного обеспечения. Бизнес-клиенты стараются изо всех сил, чтобы полностью определить эти требования, но постоянно меняющийся рынок, их недостаточное понимание того, что им действительно нужно, и трудности в полном описании ожидаемой системы до ее реализации делают изменения требований практически неизбежными.

2. Технология работает без каких-либо проблем. Все технологии, используемые при разработке программного обеспечения, должны надежно работать, как и планировалось изначально. К несчастью, в проект часто включают такие технологии, которые до того не использовались целиком, в комбинации или для тех же целей. Более того, технологические стандарты иногда меняются прямо во время разработки проекта.

3. Люди должны быть предсказуемы и надежны, как машины. План требует выполнения специфической сети задач, каждая из которых требует определенного количества часов от сотрудника, имеющего специальные навыки, которому даны четко определенные исходные данные. К сожалению, сеть задач меняется с каждым изменением требований. Еще бо́льшая проблема – то, что люди не машины. У них бывают хорошие и плохие дни, разный уровень профессионализма, отношение к делу и умственные способности. В итоге задачи выполняются не так, как изначально планировалось.

Индустрия разработки программного обеспечения понимает эти трудности и годами пыталась решить их путем наращивания усилий по планированию. Проект планирования мог занимать столько же времени, сколько и сам девелопмент. Подготовительный этап стал включать в себя сбор требований, определение архитектуры и детальную разработку планов.

Но весь этот труд был полезен, только если план основывался на точной информации и не изменялся в течение времени. Метод эффективен, когда задача хорошо понятна и относительно стабильна, а план, соответственно, остается неизменным. Если это не так, то предиктивный процесс терпит неудачу. Он не приспособлен к тому, чтобы справляться с неизвестным и неожиданным, его возможности по решению проблем ограниченны.

Множество традиционных производителей успешно используют модель прогнозируемого процесса. Выигрыш такого метода в повторном выполнении разработанного плана, создании машины за машиной или тостера за тостером. В девелопменте программного обеспечения подобного преимущества нет, план его разработки выполняется только один раз. Именно то, что делает предиктивные процессы подходящими для производства, где единственный цикл производственного процесса создает большой объем продукта, плохо подходит для программного обеспечения, где один цикл разработки предоставляет только один продукт.

Полезный инструмент, оценивающий уверенность и предсказуемость проекта, – график Стейси[2]. График Стейси измеряет уверенность в сравнении с непредсказуемостью различных аспектов работы и определяет, где план будет провален.

Мы использовали его, чтобы смоделировать три аспекта в разработке программного обеспечения: требования, технологии и люди, как показано на рис. 1.3.

[2] *R. Stacey*, Complexity and Emergence in Organizations (London: Routledge, 2001).

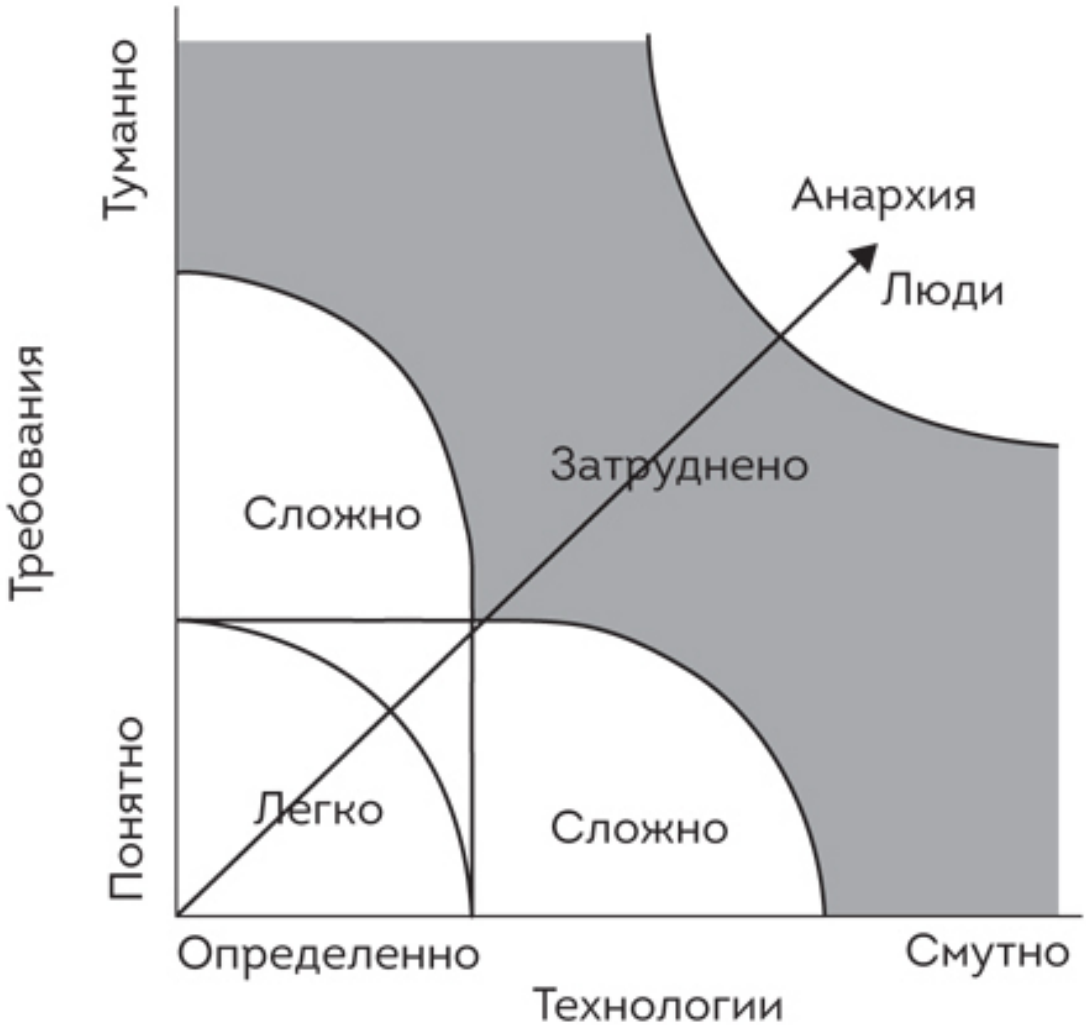

Рис. 1.3. График Стейси

Можно изобразить проекты по разработке программного обеспечения так, как показано ниже.

Требования: от близко к определенности без риска изменений до далеко от определенности со смутными, сложными характеристиками и множеством ожидаемых изменений.

Технологии: от хорошо известных и понятных до неопределенных, когда разработка и используемые технологии обычно состоят из множества продуктов, взаимодействующих через интерфейсы на различных уровнях разработки и выпуска программного обеспечения.

Люди: от знакомых и постоянных, когда разработкой занимается одна команда с малым количеством людей, до проектов, включающих больше чем четыре-пять человек, часто сотни, которые постоянно меняются. У каждого человека своя точка зрения на тот или иной вопрос, свой характер и настроение, поэтому при взаимодействии в группах или командах непредсказуемость результатов работы является значительной.

Используя график Стейси, можно увидеть, что проекты по разработке программного обеспечения как минимум сложные, а иногда и хаотичные. Предиктивный процесс, на котором основаны каскадный и традиционный методы девелопмента софта, пригоден только для простой, повторяющейся работы. Вы можете определить, правильный ли процесс вы используете, по ставке доходности, степени успеха. Если бы предиктивный подход был под-

ходящим для проектов разработки программного обеспечения, ставка доходности (или процент успешного завершения проектов) была бы очень высокой – около 99,99 %. Однако рассмотренный ранее отчет The Standish Group оценивает процент успеха проектов разработки программного обеспечения, использовавших предиктивный метод, в 14 %.

Предиктивный процесс не подходит для решения проблем в девелопменте софта. Разработка программного обеспечения – нелегкая задача, и построение ее на предиктивном процессе приведет к неудаче. Наши доказательства основаны на повышении процента успешности проектов, в которых стал применяться Scrum-метод.

Люди иногда сравнивают разработку программного обеспечения со строительством мостов. Такие инженерные дисциплины находятся на графике Стейси где-то посередине между простыми и сложными. Стандартизация относит эту работу к сложным. Существует три формы стандартизации. Во-первых, есть законы Ньютона, объясняющие, как физические объекты взаимодействуют друг с другом. Во-вторых, применяются стандартные материалы, такие как деревянный брус, стальная арматура, крепления с известными размерами и характеристиками. В-третьих, есть различного рода стандарты, описанные в разных документах и проверяемые различными органами. Ничего из этого не существует в программном обеспечении. Более того, при таком быстром развитии технологий, как сейчас, это вряд ли изменится.

Пример: Parametric Technology Corporation

Parametric Technology Corporation (PTC) – международная компания, насчитывающая около 5000 сотрудников, которая занимается разработкой программного обеспечения управления жизненным циклом продуктов. Эти продукты, которые выросли из CAD/CAM (систем автоматизированного проектирования/производства), помогают некоторым крупнейшим в мире организациям, например Raytheon, BAE System, Airbus, управлять разработкой массивных систем, таких как «Аэробус-А380». Они делают это отчасти путем отслеживания конфигурации всех частей, узлов и сборок.

В 2005 году компания PTC страдала от всех симптомов предиктивного процесса разработки программного обеспечении.

1. Выпуск новой версии продукта занимал все больше и больше времени. Выпуск релизов сползал с 18 месяцев к 24, и казалось, что текущий релиз опять займет больше времени.

2. График разработки релиза не выполнялся. Сдвиг от первоначального графика уже составлял девять месяцев и увеличивался шаг за шагом. Потребители, которые полагались на этот график, были не в восторге.

3. Стабилизация продукта перед релизом также занимала все больше и больше времени. Стабилизация была причиной как минимум двух третей задержек времени.

4. Планирование занимало все больше и больше времени и было неправильным. До шести месяцев занимало планирование следующей версии продукта, и даже в этом случае план был неправильным и требовал внесения изменений.

5. Внести изменения в середине разработки оказывалось проблематично. Было трудно сказать, сдвиг графика, процесс стабилизации или проблемы качества становились причиной необходимости внесения изменений, но определенно имелась необходимость в ряде важных изменений.

6. Качество ухудшалось. Это было серьезной и усиливающейся проблемой.

7. Авралы наносили моральный вред. PTC испытывала проблемы с наймом хороших специалистов.

В PTC использовали в процессе разработки каскадный процесс и, для того чтобы он работал лучше, пытались зафиксировать все требования. Требования, касающиеся функционала, и требуемые спецификации собирались в исчерпывающий документ. Только после того как финальные требования были утверждены, они передавались разработчику. Пока формулировались требования, разработчики либо исправляли текущие обнаруженные ошибки, либо просто скучали без дела. Персоналу по качеству не позволялось начинать тестирование, пока продукт не был полностью закончен. Таким образом, у них оставалось меньше времени на выполнение работы. Затем они были вынуждены выпускать недостаточно хорошо протестированный продукт, поскольку приближалась дата релиза.

Джейн Вачутка, вице-президент по разработке продукта Windchill в компании PTC, как новый сотрудник попыталась применять используемый в компании каскадный метод и столкнулась со всеми обычными для этого метода проблемами. На предыдущей работе она использовала множество некаскадных процессов, подобных тем, что помогли достигнуть успеха проекту ФБР «Страж». В соответствии с этим методом проект состоит из одного или нескольких повторений работы (итераций), каждый из которых длится не более 30 дней. Множество небольших команд разработчиков выбирают наиболее важные требования для каждой итерации и превращают их в часть готового к употреблению программного обеспечения. Все эти части затем объединяются в одну полностью законченную и готовую к применению программу. В конце каждой последующей итерации другие части и надстройки программы добавляются к существующему функционалу.

Брайан Шепард, исполнительный вице-президент по развитию продуктов компании PTC, был настроен скептически, когда в 2007 году Джейн предложила новый процесс производства программного обеспечения.

Она просила позволить ей раньше подключать программистов к разработке, задействовать контроль качества и не выпускать продукты, которые не прошли полного тестирования. Джейн подчеркнула, что функциональные характеристики могут быть несовершенными, так как группа управления разработкой часто будет получать и использовать части разрабатываемого продукта в течение всего цикла разработки, чтобы давать обратную связь. Брайан согласился попробовать новый процесс – гибкий метод разработки под названием Scrum. При этом он предупредил Джейн: «У тебя нет права на ошибку!»

Когда Джейн сообщила своим сотрудникам о новом методе, они также отнеслись к этой новости скептически и не сразу вникли в суть нового метода, по-прежнему изо всех сил пытаясь добиться совершенных результатов на каждом этапе разработки, стараясь уточнять, что они делают именно то, что другие хотели. Тем не менее, по мере того как менеджеры проектов накапливали опыт в использовании нового метода, они больше не старались получить идеальные функциональные характеристики перед передачей в разработку, так как дополнительные характеристики добавляются в течение всего времени до выпуска релиза. Поскольку теперь PTC разрабатывала полностью функционирующее программное обеспечение каждые 30 дней, ее разработчики смогли напрямую сотрудничать с клиентами в рамках каждой итерации. Разработчики получили понимание требований и то, как эти требования могут быть реализованы лучше всего. Клиенты заметили изменения и начали работать с командами разработчиков в течение каждой итерации. Клиенты помогали авторам создавать функционал и получали именно то, что хотели.

Команда управления продуктами имела в своем распоряжении трех-, двух- и однолетний набор требований. За три года до выпуска это было всего лишь видение продукта с описанием самых главных возможностей. Более детальная картина будущего продукта прорабатывалась во второй год разработки. Для текущего года 30-дневные итерации определялись на первые шесть месяцев и составлялась дорожная карта на последующие шесть. Набор требований для каждого года имел больше деталей, чем предыдущий. Разработчики труди-

лись над набором требований для одного года совместно с клиентами PTC, чтобы выяснить больше деталей. Вся организация стала средоточием творчества и продуктивности.

В течение двух лет все обязательства Джейн перед Брайаном были выполнены. Джейн и ее сотрудники выпускали программное обеспечение каждые 12 месяцев, в то время как ранее выпуск занимал 24 месяца. Продукт был высокого качества. К 2011 году PTC изменилась – она превратилась в прозрачную организацию как внутри, так и для потребителей. Сюрпризы случались редко, клиенты знали, чего и когда ждать. Дефекты были редкими и к 2012 году практически равнялись нулю. Новые детали, пользовательские интерфейсы и возможности рабочего потока были добавлены. Продукт был переработан, стал безопасным от внешних угроз. В итоге бюджет и организационные расходы упали каждый более чем на 10 %. По поручению Брайана Шепарда для организации по разработке программного обеспечения было построено новое здание, которое отражало прозрачность, столь важную для перемен: все располагались в едином пространстве, без отдельных кабинетов, и стены были из стекла.

Недавно Джим Хеппermann, CEO PTC, во время обсуждения с менеджерами вопроса о повышении бюджетов подразделений попросил их всех поблагодарить Джейн за снижение расходов при одновременном повышении качества и функционала. Сэкономленные ее усилиями средства он может разделить между всеми подразделениями компании.

Однажды Джейн и Джим присутствовали на конференц-связи с компанией в Израиле, оценивающей возможность применения продуктов PTC. Джейн сказала CEO, что компания Raytheon использует продукты PTC по всему миру, и просила связаться с их руководством. Она знала, что в Raytheon не только впечатлены продуктами PTC, но и вдохновлены тем, что новый процесс PTC по разработке программ исключает сюрпризы и есть возможность корректировать свои графики с PTC в режиме реального времени. Компания Raytheon даже переняла у PTC процесс разработки программного обеспечения. Джим сообщил потенциальным клиентам, что последний релиз – самый красивый продукт, когда-либо выпущенный PTC, в первую очередь потому что Джейн изменила процесс разработки.

Итог

Ранее разработка программного обеспечения была скорее неудачной, в первую очередь по причине использования предиктивных процессов для комплексных работ. Переход к Scrum, эмпирическому процессу, значительно увеличивает шансы на успешное завершение проекта разработки программного обеспечения.

Можно получить функции программного обеспечения, готовые к использованию, в течение 30 дней или даже меньше. Не позволяйте вашим разработчикам убедить вас в обратном, потому что сотни тысяч их коллег делают это с 2000-х годов. Программный продукт может быть большим, но его можно собрать фрагмент за фрагментом – каждый за 30 дней.

2. Scrum: правильный процесс приводит к правильному результату

В предыдущей главе мы узнали, что эмпирический процесс – правильный для девелопмента программного обеспечения. Теперь давайте посмотрим, как эмпиризм работает и как можно с его помощью создать ПО. Мы изучим эмпиризм через призму гибкого процесса разработки программного обеспечения, который мы назвали Scrum.

Конец ознакомительного фрагмента.