ЧИСТЫЙ AGILE

ОСНОВЫ ГИБКОСТИ

РОБЕРТ МАРТИН

Библиотека программиста (Питер)

Роберт Мартин

Чистый Agile. Основы гибкости

«Питер»
2020

УДК 004.3
ББК 32.973.2-018

Мартин Р. С.

Чистый Agile. Основы гибкости / Р. С. Мартин — «Питер», 2020 — (Библиотека программиста (Питер))

ISBN 978-5-4461-1552-5

Прошло почти двадцать лет с тех пор как появился Манифест Agile. Легендарный Роберт Мартин (Дядя Боб) понял, что пора стряхнуть пыль с принципов Agile, и заново рассказать о гибком подходе не только новому поколению программистов, но и специалистам из других отраслей. Автор полюбившихся айтишникам книг «Чистый код», «Идеальный программист», «Чистая архитектура» стоял у истоков Agile. «Чистый Agile» устраняет недопонимание и путаницу, которые за годы существования Agile усложнили его применение по сравнению с изначальным замыслом. По сути Agile – это всего лишь небольшая подборка методов и инструментов, помогающая небольшим командам программистов управлять небольшими проектами, … но приводящая к большим результатам, потому что каждый крупный проект состоит из огромного количества кирпичиков. Пять десятков лет работы с проектами всех мыслимых видов и размеров позволяют Дяде Бобу показать, как на самом деле должен работать Agile. Если вы хотите понять преимущества Agile, не ищите лёгких путей – нужно правильно применять Agile. «Чистый Agile» расскажет, как это делать разработчикам, тестировщикам, руководителям, менеджерам проектов и клиентам.

УДК 004.3
ББК 32.973.2-018

ISBN 978-5-4461-1552-5

Содержание

Роберт Мартин
Чистый Agile. Основы гибкости

Robert C. Martin
Clean Agile. Back to Basics

* * *

Отзывы на книгу «Чистый Agile»

Проделав огромный путь, Agile, не без помощи того самого Дяди Боба, достиг вершин. Безусловно, эта история знает свои взлеты и падения. Восхитительная книга, которую вы сейчас держите в руках, сочетает в себе историческую хронику и мемуары. Вся мудрость Agile собрана здесь. Если вам интересно, что такое Agile и как он появился, – эта книга для вас.

Гради Буч

Разочарования Дяди Боба пронизывают каждую строку «Чистого Agile», но, поверьте, эти разочарования оправданны. Все, что создано в мире Agile, – просто капля в море по сравнению с тем, что еще можно из него сотворить. Эта книга – взгляд Дяди Боба на перспективу. Его вклад велик. Его стоит послушать.

Кент Бек

Полезно почитать о том, что Дядя Боб думает об Agile. Может быть, вы новичок, а может, уже матерый спец – в любом случае вы найдете в этой книге что-то для себя. Я подписываюсь почти под каждым словом в ней. В некоторых главах я вижу свои собственные недостатки, что греха таить. Теперь я проверяю наш код дважды, а это 85,09 %.

Джон Керн

В этой книге, словно под увеличительным стеклом, можно в подробностях разглядеть методологию Agile. Дядя Боб, несомненно, один из умнейших людей, которых я знаю. Его рвение к программированию бесконечно. Он как никто другой может развеять мистический туман, сгустившийся над Agile.

Из предисловия Джерри Фицпатрика

Всем программистам, кто хоть раз боролся с ветряными мельницами или водопадами

Предисловие

Так что же такое методология гибкой разработки Agile? Как появилась на свет? Как эволюционировала?

В этой книге Дядя Боб дает глубокомысленные ответы на эти вопросы. А еще рассказывает о разных способах неправильного или искаженного понимания методологии Agile. Его взгляды очень важны, поскольку он авторитет в этой области. Ведь именно с его именем связано появление Agile.

Мы дружим с Бобом уже не один год. Впервые я встретил его в 1979-м, когда устроился на работу в отдел телекоммуникаций компании Teradyne. Я был инженером-электриком, моя работа заключалась в том, что я помогал устанавливать и обслуживать нашу продукцию. Позже я вырос до разработчика аппаратных средств.

Где-то спустя год моей работы в компании стали искать новые идеи для своей продукции. В 1981-м мы с Бобом выдвинули идею создания электронного телефонного администратора – по сути дела, он представлял собой службу голосовой почты с функцией переадресации вызовов. В компании нашу концепцию приняли с одобрением, и вскоре мы приступили к разработке E.R. – электронного администратора (The Electronic Receptionist). Прототип получился шедевральным. Он работал под управлением операционной системы MP/M на базе процессора Intel 8086. Голосовые сообщения хранились на пятимегабайтовом винчестере Seagate ST-506. Я занимался разработкой порта для передачи голоса, а Боб писал приложение. Закончив разработку, я тоже написал часть кода приложения. Поэтому с тех пор я еще и разработчик.

То ли в 1985-м, то ли в 1986-м Teradyne неожиданно остановила разработку E.R. и без нашего ведома отозвала заявку на патент. Компания вскоре пожалела о принятом решении, а мы с Бобом жалеем и по сей день.

В конце концов мы ушли из Teradyne в поисках лучшей доли. Боб занялся консалтинговым бизнесом в районе Чикаго. Я же стал преподавателем и ударился в разработку программ. Мы умудрялись не терять друг друга из виду даже после моего переезда в другой штат.

К 2000 году я преподавал объектно-ориентированный анализ и проектирование в Learning Tree International. Курс включал в себя преподавание UML и унифицированный процесс разработки (USDP). Тогда я уже поднаторел в этих технологиях, однако ничего не понимал в Scrum, экстремальном программировании и прочих методиках.

А в феврале 2001 года появился Манифест Agile. Моя реакция была примерно как у всех: «Что еще за Agile?» Единственный манифест, о котором я знал, – это Манифест Коммунистической партии, составленный Карлом Марксом и Фридрихом Энгельсом. Этот Agile призывал взяться за оружие? Чертовы айтишные радикалы!

Манифест породил массы бунтарей. Он вдохновил программистов на создание лаконичного чистого кода посредством совместной адаптивной работы с отлаженной обратной связью. Они выглядели заманчивой альтернативой тяжеловесным процессам вроде каскадной модели и USDP.

Прошло уже 18 лет, как был обнародован Манифест. Для большинства современных разработчиков все это история древнего мира. Именно поэтому ваше понимание Agile может отличаться от представлений его основателей.

Эта книга ставит своей целью точно передать посыл идеи. В этой книге, словно под увеличительным стеклом, можно в подробностях разглядеть методологию Agile. Дядя Боб, несомненно, один из умнейших людей, которых я знаю. Его рвение к программированию бесконечно. Он как никто другой может развеять мистический туман, сгустившийся над Agile.

Джерри Фицпатрик Software Renovation Corporation Март 2019 года

Введение

Книга, которую вы держите в руках, – не научное исследование. Я не старался провести тщательный обзор литературы. То, что вы собираетесь прочесть, – мои личные воспоминания, наблюдения и мнения. Как-никак, я около двадцати лет связан с Agile.

Книга написана в неформально-разговорном стиле. Поэтому иногда я могу выбирать не очень корректные выражения. Сам я далеко не любитель крепкого словца, правда одно (несколько искаженное) бранное слово попало на эти страницы, но лишь потому, что иначе выразиться было никак!

Я бы не сказал, что книга гневлива. Когда вдруг меня прошибало, что нужно как-то обосновать написанное, я приводил источники, на которые ссылался. Я сверялся с мнениями других ребят из сообщества Agile, которые в деле столько же времени, сколько и я. Иногда даже нарочно просил некоторых дополнить книгу своим мнением или выразить свое несогласие, чему посвящены отдельные главы и разделы. Как бы то ни было, не стоит воспринимать эту книгу как научный труд. Наверное, правильнее бы воспринимать эту книгу как мемуары – старческое брюзжание, адресованное новичкам в Agile, которые только делают свои первые шаги.

Книга подойдет как программистам, так и простым любителям. Это не техническая литература.

Тут нет кода. В книге дается обзор первоначальной цели разработки Agile без углубления в какие-либо технические нюансы программирования, тестирования и управления.

Книга невелика. Просто потому, что много писать и не надо. Agile – небольшая идея, предназначенная для решения небольших задач, поставленных небольшими командами программистов, которые выполняют небольшую работу. Agile не рассчитан на решение крупных задач больших команд программистов, которые занимаются крупными проектами. Есть даже что-то ироничное в том, что такое решение для таких мелких задач вообще имеет название. В конце концов, мелкие задачи, о которых тут говорится, решили еще в 1950-е и 60-е, почти сразу, как изобрели программное обеспечение в принципе. Даже в те далекие времена небольшие команды научились неплохо справляться с небольшим объемом работ. Однако все испортилось в 1970-х. Тогда маленькие команды разработчиков, выполняющие небольшие объемы работ, запутались в идеях, пропагандирующих выполнение крупных работ в крупных командах.

А разве не так? Боже, да не так! Крупная работа не выполняется большими командами. На самом деле крупная работа выполняется большим количеством маленьких команд, которые в свою очередь выполняют много небольших задач. В 1950-х и 60-х годах программисты понимали это на уровне инстинкта. Но в 1970-е годы про это просто-напросто забыли.

А почему забыли? Полагаю, что так произошло из-за неравномерности. Количество программистов в мире стало резко расти в 1970-х годах. До этого в мире было всего несколько тысяч программистов. Но потом их количество резко выросло до сотен тысяч. Сейчас их число достигает сотни миллионов.

Те самые программисты 1950-х и 60-х годов были взрослыми. Они стали программистами лет в 30, 40 или даже в 50. К 1970-м годам, когда программистов вдруг стало тьма-тьмущая, эти «старички» ушли на пенсию. Поэтому некому было обучать новое поколение программистов. Молодые программисты от 20 лет и старше начали работать как раз тогда, когда более опытные ребята уже начали уходить, поэтому их опыт не передавался эффективно.

Некоторые скажут, что с этого события в программировании началось смутное время. На протяжении 30 лет мы боролись за идею, которая гласила, что следует выполнять большую работу в больших командах, совершенно не осознавая, что секрет успеха – это выполнение большого количества мелких задач множеством небольших команд.

Затем, в середине 1990-х, наконец было переосмыслено то, что было упущено. Идея работы в небольших командах получила новую жизнь. Эта идея распространилась в сообществе разработчиков программного обеспечения и набирала обороты. В 2000-х мы поняли наконец, что нужно перезагрузить всю отрасль целиком.

Теперь нам нужно вспомнить то, что знали те, кто были до нас, на уровне инстинкта. Нам еще раз потребовалось понять, что крупные задачи выполняются небольшими командами, которые сотрудничают между собой в решении небольших задач. Мы подумали, что идея, у которой есть имя, больше привлечет внимания. И мы назвали ее *Agile*.

Я написал это введение в самом начале 2019-го. Прошло уже около двух десятилетий с перезагрузки 2000-х годов, и, кажется, пришло время для еще одной.

Почему? Да потому, что простое и маленькое послание Agile за эти годы потеряло свою суть. Его перемешали с концепциями Lean, Kanban, LeSS, SAFe, Modern, Skilled и многими другими. Перечисленные идеи тоже по-своему хороши, но это все равно не Agile.

Вот и пришло время, чтобы напомнить нам то, о чем знали наши предки в 1950-х и 60-х годах, и о том, что мы вновь усвоили в начале 2000-х. Пора вспомнить, что такое Agile на самом деле.

В этой книге вы не найдете ничего особенно нового, поражающего или изумляющего. Никаких революций, ломающих привычные шаблоны. То, что вы узнаете отсюда об Agile, – это то, о чем уже говорилось в 2000-х. Хотя нет. Тут говорится с другой точки зрения. Ведь за

20 лет, которые прошли за это время, мы усвоили что-то новое, и это включено в книгу. Но в целом посыл этой книги тот же, что и в 2001-м, и в 1950-м.

Посыл стар как мир. Но тем не менее это истина. Этот посыл предлагает нам небольшую идею для решения небольших задач, поставленных небольшими командами программистов, которые выполняют небольшую работу.

Благодарности

Первая моя благодарность – двум отважным программистам, которые не без удовольствия открыли (а может, и заново открыли) методы, изложенные в этой книге, – Уорду Каннингему и Кенту Беку.

Следующую благодарность выражаю Мартину Фаулеру. Без его твердой руки революция, произведенная Agile, могла так и не увидеть свет.

Кен Швабер заслуживает особого упоминания за его неукротимую энергию в продвижении и внедрении Agile.

Мэри Поппендик также заслуживает отдельного упоминания за самоотверженность и неиссякаемую энергию, которую она вкладывала в движение Agile, и ее заботу об Agile Alliance.

На мой взгляд, Рон Джеффрис благодаря своим выступлениям, статьям, блогам и теплоте своего характера может считать себя совестью в начале движения Agile.

Майк Бидл отлично отстаивал честь Agile, однако погиб ни за что от рук бездомного на улицах Чикаго.

Прочим авторам оригинального Манифеста Agile здесь также отводится отдельное место. Перечислю их: Ари ван Беннекум, Алистер Кокберн, Джеймс Греннинг, Джим Хайсмит, Эндрю Хант, Джон Керн, Брайан Марик, Стив Меллор, Джефф Сазерленд и Дейв Томас.

Джим Ньюкирк, мой друг и деловой партнер, в то время без устали работал в поддержку Agile вопреки личным трудностям, которые большинство из нас (и я в том числе, безусловно) не могут даже представить.

Также я хочу упомянуть людей, работавших в корпорации Object Mentor Inc. Они приняли на себя основные риски по внедрению и продвижению Agile. Многие из них присутствуют ниже на фото, которое было сделано в начале первых уроков курса XP Immersion.

Задний ряд: Рон Джеффрис, я (автор), Брайан Баттон, Лоуэлл Линдстрем, Кент Бек, Мика Мартин, Анжелика Мартин, Сьюзен Россо, Джеймс Греннинг. Передний ряд: Дэвид Фарбер, Эрик Мид, Майк Хилл, Крис Бигей, Алан Фрэнсис, Дженнифер Конке, Талиша Джеффер-

сон, Паскаль Рой. Не присутствуют: Тим Оттингер, Джефф Лэнгр, Боб Косс, Джим Ньюкирк, Майкл Фезерс, Дин Уэмплер и Дэвид Хелимски

Также я хочу упомянуть ребят, которые смогли собраться и сформировать альянс Agile Alliance. Некоторых из них можно увидеть ниже на фото – оно было сделано в начале заседания альянса в нынешнем августе.

Слева направо: Мэри Поппендик, Кен Швабер, автор, Майк Бидл, Джим Хайсмит (не присутствует: Рон Крокер)

Наконец, спасибо всем ребятам из Pearson, в особенности моему издателю Джули Файфер.

Об авторе

Роберт С. Мартин (Дядя Боб) является практикующим программистом с 1970 года. Он является также соучредителем cleancoders.com, где представлены различные видеоуроки для разработчиков программного обеспечения, учредителем компании Uncle Bob Consulting LLC, оказывающей услуги по консультированию, подготовке и развитию навыков крупным корпорациям по всему миру. Был высококлассным специалистом в консалтинговой компании, занимающейся отраслью программного обеспечения, 8th Light Inc., расположенной в Чикаго.

Боб Мартин написал десятки статей для различных профессиональных журналов и систематически выступает на международных конференциях и выставках. Он также является создателем известных образовательных видео на cleancoders.com. Боб Мартин является автором и редактором многих книг, в том числе:

«Разработка объектно-ориентированных приложений на C++ по методу Буча» (Designing Object-Oriented C++ Applications Using the Booch Method)

«Языки паттернов в процессе создания программ 3» (Patterns Languages of Program Design 3)

«Еще больше сокровищ C++» (More C++ Gems)

«Экстремальное программирование на практике» (Extreme Programming in Practice)

«Быстрая разработка программ. Принципы, примеры, практика» (Agile Software Development: Principles, Patterns, and Practices)

«UML для программистов на Java» (UML for Java Programmers)

«Чистый код»[1]

«Идеальный программист»[2]

«Чистая архитектура»[3]

Лидер в отрасли разработки программного обеспечения, г-н Мартин три года был главным редактором журнала C++ Report, а также первым председателем Agile Alliance.

---

[1] *Мартин Р.* Чистый код: создание, анализ и рефакторинг. – СПб.: Питер, 2018. – 464 с.: ил.

[2] *Мартин Р.* Идеальный программист. Как стать профессионалом разработки ПО. – СПб.: Питер, 2019. – 224 с.: ил.

[3] *Мартин Р.* Чистая архитектура. Искусство разработки программного обеспечения. – СПб.: Питер, 2020. – 352 с.: ил.

От издательства

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

Мы будем рады узнать ваше мнение!

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

Глава 1. Введение в Agile

В феврале 2001-го группа из семнадцати экспертов в области разработки программного обеспечения собралась в городке Сноуберд, штат Юта. На собрании обсуждалось плачевное состояние отрасли. Тогда большинство программ создавалось посредством неэффективных тяжеловесных фреймворков с большим количеством ритуалов, наподобие каскадной модели и раздутых реализаций Rational Unified Process (RUP). Целью этих экспертов было создание манифеста, который провозглашал бы более эффективный и легковесный подход.

Нельзя сказать, что царило единодушие. У всех семнадцати был разный опыт и, соответственно, сильные расхождения во мнениях. Рассчитывать, что такое собрание быстро придет к общему мнению, было бы слишком наивно. И все же, несмотря на все трудности, согласие было достигнуто, и Манифест Agile увидел свет – так родилось одно из самых мощных и живучих движений в отрасли программного обеспечения.

В этой отрасли практически все движения происходят по одному и тому же пути. Поначалу есть меньшинство, восторженно выступающее в поддержку чего-либо, другое меньшинство заряженных критиков и подавляющее большинство – те, кому, мягко говоря, нет дела.

Многие из этих движений хиреют или никогда не проходят этот этап. Можно вспомнить аспектно-ориентированное программирование, логическое программирование или CRC-карты. Некоторые, однако, способны преодолеть такую пропасть, становясь чрезвычайно популярными и разносторонними. Некоторым удается оставить позади противоречия и занять господствующее положение в современной мысли. Объектно-ориентированное мышление можно считать примером последнего случая. И Agile тоже.

К сожалению, как только движение набирает последователей и начинает распространяться, то сталкивается с искажением и незаконным присваиванием. Продукция и методики, не имеющие ничего общего с оригинальным движением, присваивают чужое имя, чтобы нажиться на его славе и значимости. Так случилось и с Agile.

Эта книга, написанная спустя почти два десятка лет после событий в Сноуберде, ставит своей целью точно передать посыл идеи. Эта книга – попытка в высшей мере кратко и точно описать Agile без брехни и обиняков.

В ней представлены основные принципы Agile. В приукрашивании и расширении этих идей нет ничего плохого. Тем не менее производные от Agile – это уже не сам Agile. Это дополненный Agile со своими опциями. А то, о чем вы узнаете из этой книги, – это и есть тот самый чистый Agile, который всегда был и непременно будет.

История Agile

Когда зародился Agile? Вероятно, более 50 тысяч лет назад, когда люди впервые решили работать совместно ради общей цели. Идея постановки небольших промежуточных целей и измерения продвижения после их достижения у человека проявляется на подсознательном уровне, поэтому вряд ли это какая-то настоящая революция.

Когда впервые появился Agile в современном мире? Трудно сказать. В моем представлении, первый паровой двигатель, первая мельница, первый двигатель внутреннего сгорания, первый самолет были созданы с помощью методик, которые сейчас можно отнести к Agile. Я считаю так, потому что предпринимать небольшие измеряемые шаги очень естественно для человека, сложно представить, что это происходит иначе.

Так когда же Agile появился среди программистов? Хотел бы я быть мухой на стене у Алана Тьюринга, когда тот писал свою книгу в 1936 году[4]. По моим догадкам, многие свои «программы» он написал, разбивая работу на мелкие этапы с изобилием отладки по исходному тексту вручную.

Я также хорошо представляю, что первый код, который он написал для автоматической вычислительной машины (Automatic Computing Engine, ACE) в 1946 году, был написан постепенно, маленькими этапами, с частым проведением ручной отладки по исходному тексту и даже с небольшим тестированием в действии.

В первые дни существования программного обеспечения можно найти много примеров решения задач, которые сейчас бы отнесли к Agile. Например, программисты, писавшие код для управления пилотируемым космическим кораблем «Меркурий», работали по этапам с интервалом в полдня с перерывами на модульное тестирование.

Об этом периоде есть много материалов. Крэг Ларман и Вик Базили написали историю, которая кратко изложена в «вики» Уорда Каннингема[5], а также в книге Ларман *Agile & Iterative Development: A Manager's Guide*[6].

Однако существовал не только Agile. Действительно, есть конкурирующая методология, которая пользовалась значительным успехом в производстве и промышленности в целом: научная организация труда.

Научная организация труда – это командно-административный подход с иерархической структурой. Менеджеры применяют научную организацию труда, чтобы определять наилучший набор процедур для достижения цели и отдавать распоряжения всем подчиненным для выполнения плана с точностью до буквы. Другими словами, сначала планируется крупная задача, затем проводится тщательная и подробная проработка плана.

Научная организация труда, вероятно, такая же древняя, как пирамиды в Египте, Стоунхендж или прочие подобные работы древних времен: с трудом верится, что можно работать по-другому. Еще раз замечу, идея повторять успешный опыт настолько глубоко подсознательно заложена в человеке, что ее сложно охарактеризовать как революционную.

Научная организация труда получила свое название из работ Фредерика Уинслоу Тейлора, написанных в 1880-х. Тейлор сформировал этот подход, придав ему коммерческую ценность и сделав состояние на консультировании по управлению. Метод получил широкий успех

---

[4] *Turing A. M.* 1936. On computable numbers, with an application to the Entscheidungsproblem [доказательство]. Proceedings of the London Mathematical Society (Труды Лондонского математического общества), 2 (изд. 1937), 42(1):230–65. – Лучший способ ознакомиться с этой работой – прочитать произведение Чарльза Петцольда: *Petzold C.* The Annotated Turing: A Guided Tour through Alan Turing's Historic Paper on Computability and the Turing Machine. Indianapolis, Indiana: Wiley, 2008.

[5] «Вики» Уорда, c2.com – сайт оригинальной «Википедии», первый день появления в сети Интернет. Пусть поддержка длится как можно дольше.

[6] *Larman C.* Agile & Iterative Development: A Manager's Guide. Boston, Massachusetts: Addison-Wesley, 2004.

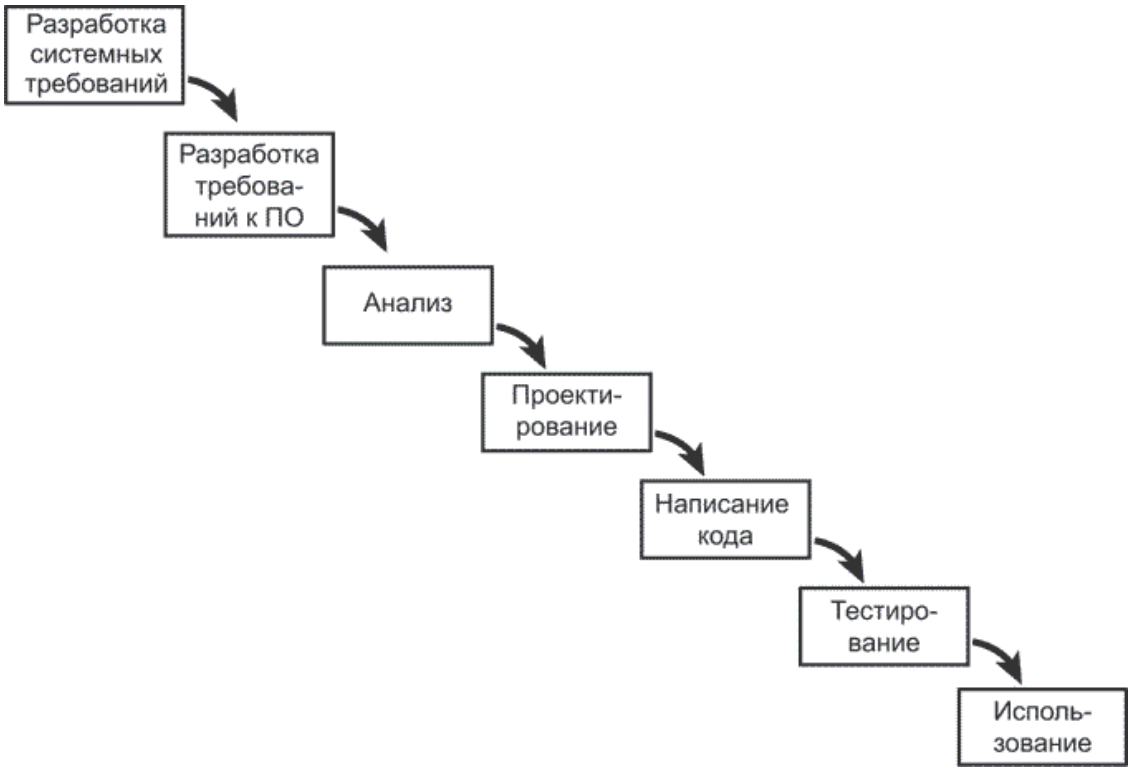
и привел к значительному повышению эффективности и производительности в последующие десятилетия.

И так случилось в 1970-е, что мир разработчиков программного обеспечения разделился на два лагеря – сторонников того или иного метода. Прото-Agile (Agile, еще не получивший название «Agile») предлагал предпринимать в зависимости от обстоятельств небольшие случайные шаги, которые можно измерить и выделить, для того чтобы идти в правильном направлении к наилучшему исходу. Научная организация труда призывала откладывать действие до тщательного анализа и проработки заранее подготовленного плана. Прото-Agile прекрасно подходил для проектов с низкой стоимостью внесения изменений, которые решали частично определенные задачи с произвольно поставленными целями.

Научная организация труда лучше всего подходила для проектов, которые решали четко определенные задачи с весьма определенными целями. И в них стоимость изменений уже возрастала. Какими были проекты в отрасли разработки программного обеспечения? Была ли в этих проектах стоимость изменений высока? Были ли задачи определены четко? Или стоимость изменений была низкой, а цели были поставлены произвольно?

Не вчитывайтесь слишком внимательно в предыдущий абзац. Насколько я знаю, никто не задавался такими вопросами. Иронично и то, что путь, выбранный в 1970-х годах, кажется, был обусловлен, скорее, волей случая.

В 1970 году Уинстон Ройс в своей работе[7] описал идеи для управления крупномасштабными проектами по разработке программного обеспечения. В этой работе присутствовала схема (рис. 1.1), которая наглядно изображала его план. Ройс не был автором этой схемы и не продвигал ее в качестве плана. В действительности график представлял собой изображение соломенного человечка и помогал Ройсу ориентироваться в последующих страницах своего труда.



[7] *Royce W. W.* 1970. Managing the development of large software systems. Proceedings, IEEE WESCON, August: 1–9. URL: http://www-scf.usc.edu/~csci201/lectures/Lecture11/royce1970.pdf.

Рис. 1.1. Схема Уинстона Ройса, которая стала источником вдохновения для разработки каскадной модели

Схема была расположена на видном месте. А учитывая, что люди делают логические выводы о содержании статьи, посмотрев схему на первой или второй странице, это привело к резкому сдвигу в отрасли программного обеспечения.

Схема, первоначально составленная Ройсом, гораздо больше напоминала ручей, стекающий вниз со скалистого хребта, чем ныне известную каскадную модель.

Каскадная модель стала логическим продолжением научной организации труда. При использовании этой модели на первый план ставится тщательный анализ, составление подробного плана, а затем уже доведение этого плана до завершения. Хотя Ройс и не рекомендовал такой подход, но именно эту концепцию вынесли из его работы. А потом эта концепция главенствовала в отрасли более трех десятков лет[8].

Как раз тогда и начинается моя история. В 1970-м мне было 18 лет, я работал программистом в компании A.S. C. Tabulating, расположенной в Лейк Блафф, Иллинойс.

У компании был компьютер IBM 360/30 с памятью на магнитных сердечниках 16 килобайт, IBM 360/40 с памятью 64 килобайта и микрокомпьютер Varian 620/f с памятью 6 килобайт. Я писал программы для семейства 360 на COBOL, PL/1, Fortran и ассемблере. Для 620/f я писал только на ассемблере.

Важно помнить, каково в то время было программистам. Мы писали код в программных формулярах с помощью карандашей. У нас были операторы, работающие за перфоратором, которые наносили программы на карты. Мы передавали тщательно выверенные перфокарты операторам ЭВМ, которые проводили компиляцию и тестирование в третью смену, поскольку днем, когда работа кипела, компьютеры были постоянно заняты. От начала написания кода до первой компиляции зачастую проходило несколько дней, каждый цикл разработки вследствие этого занимал, как правило, сутки.

Для меня 620/f выглядел несколько иначе. Эту машину выделили нашей команде, поэтому мы могли работать на ней столько, сколько вздумается. Мы проводили два, три, иногда даже четыре цикла разработки и тестирования за сутки. Вместе со мной в команде были люди, которые, в отличие от большинства программистов того времени, умели печатать. Поэтому мы могли штамповать свои собственные колоды перфокарт, а не зависеть от капризов операторов, работающих за перфоратором.

Какую методологию мы использовали на протяжении того времени? Это, конечно, была не каскадная модель. У нас не было концепций или подробных планов. Мы просто писали код каждый день, компилировали, тестировали и устраняли ошибки. Это был бесконечный цикл без структуры. Также это был не Agile и даже не прото-Agile. В ходе работ мы не придерживались каких-либо правил организации. Тогда не было каких-либо пакетов программ для тестирования и измеряемых временных интервалов. Просто надо было писать код и фиксить баги. День за днем, месяц за месяцем.

Впервые я узнал о каскадной модели из профессиональных журналов около 1972 года. Мне она казалась даром свыше. Неужели мы могли бы проанализировать задачу, потом предложить ее решение, а затем реализовать замысел? Реально ли было на самом деле разработать график, основанный на трех перечисленных этапах?

Неужели, когда выполнен анализ, проект продвинется вперед на треть? Я почувствовал силу этой концепции. Я хотел в это верить. Если идея сработает, то мечта воплотится.

[8] Стоит отметить, что мое толкование этой временной шкалы подвергли сомнениям. См.: *Bossavit L.* The Leprechauns of Software Engineering: How Folklore Turns into Fact and What to Do About It, Ch. 7. Leanpub, 2012.

Судя по всему, я был не один, потому что многие другие программисты и центры программирования тоже вошли в кураж. И, как я уже писал, в нашем мышлении начала преобладать каскадная модель.

Она преобладала, но не работала. В течение последующих тридцати лет мои коллеги, я, братья и сестры по программированию по всему миру неустанно старались получить право на проведение анализа и проектирование. Но каждый раз, когда мы думали, что получали желаемое, оно ускользало из наших рук на этапе реализации. Месяцы тщательного планирования пошли прахом из-за необходимости сделать безумный рывок, в итоге мы сорвали сроки под свирепыми взглядами менеджеров и заказчиков.

Несмотря на практически нескончаемый поток неудач, мы все равно настаивали на состоятельности каскадной модели. В конце концов, почему возникали неудачи? Почему тщательный анализ задачи, внимательное проектирование решения и последующая реализация нескончаемо терпят зрелищный крах? Никто даже подумать не мог, что дело было в самой стратегии. Задача должна была лечь на наши плечи. Как бы то ни было, что-то мы делали не так.

Чтобы увидеть, насколько каскадная модель захватила наши умы, посмотрите на программные языки того времени. Когда Дейкстра в 1968 году представил структурное программирование, структурный анализ[9] и структурный дизайн[10] не сильно отставали. В 1988 году, когда объектно-ориентированное программирование (ООП) набрало популярность, объектно-ориентированный анализ[11] и объектно-ориентированное проектирование[12] также не сильно отставали. Эта тройка идей, эти три этапа словно держали нас в плену. Мы просто не могли представить, что можно работать как-то по-другому.

А потом оказалось, что можно.

Зародыши преобразований, связанных с Agile, появились в конце 1980-х или в начале 90-х. В сообществе Smalltalk их признаки начали проявляться в 1980-х. В книге Буча по объектно-ориентированному проектированию, вышедшей в 1991-м, были намеки на них. Прочие решения возникли в 1991 г. в книге Кокберна Crystal Methods. Сообщество Design Patterns начало обсуждать это в 1994-м под влиянием статьи, написанной Джеймсом Коплиеном[13].

К 1995 году Бидл[14], Девос, Шэрон, Швобер и Сазерленд написали свои знаменитые труды о Scrum (Скрам)[15]. И затворы открылись. На бастионе каскадной модели образовалась брешь, и пути назад не было.

И здесь я снова возвращаюсь к нашей истории. То, что я расскажу дальше, – мои личные воспоминания, я не сверял их ни с кем из современников, участников событий. Поэтому следует предположить, что в моих воспоминаниях много опущений, недостоверностей или изложены они ужасно беспорядочно. Но не переживайте, я по крайней мере постарался рассказать все занимательно.

В первый раз мы встретились с Кентом Беком в 1994 году на той самой конференции PLoP[16], когда Коплин представил свою работу. Это была неформальная встреча, которая тол-

---

[9] *DeMarco T.* Structured Analysis and System Specification. Upper Saddle River, New Jersey: Yourdon Press, 1979.

[10] *Page-Jones M.* The Practical Guide to Structured Systems Design. Englewood Cliffs, New Jersey: Yourdon Press, 1980.

[11] *Coad P., Yourdon E.* Object-Oriented Analysis. Englewood Cliffs, New Jersey: Yourdon Press, 1990.

[12] *Booch G.* Object Oriented Design with Applications. Redwood City, California: Benjamin-Cummings Publishing Co., 1991.

[13] *Coplien J. O.* A generative development-process pattern language. Pattern Languages of Program Design. Reading, Massachusetts: Addison-Wesley, 1995. P. 183.

[14] Майк Бидл был убит 23 марта 2018 года в Чикаго психически нездоровым человеком, до этого арестованным и отпущенным 99 раз, который должен был находиться в психбольнице. Майк Бидл был моим другом.

[15] *Beedle M., Devos M., Sharon Y., Schwaber K., Sutherland J.* SCRUM: An extension pattern language for hyperproductive software development. Ссылка: http://jeffsutherland.org/scrum/scrum_plop.pdf.

[16] Pattern Languages of programming – конференция, которую проводили в 1990-х неподалеку от университета штата Иллинойс.

ком ничего не принесла. В следующий раз я встретил его в феврале 1999-го в Мюнхене на конференции, посвященной ООП. Но к тому времени я уже знал о нем намного больше.

В то время я занимался консультированием по C++ и объектно-ориентированному проектированию, летал с места на место, помогал разрабатывать и реализовывать приложения на C++ с помощью методик объектно-ориентированного проектирования.

Клиенты стали расспрашивать меня о процессе. Они слышали, что каскадная модель не применяется в объектно-ориентированном проектировании, и хотели услышать от меня совет. Я согласился с ними[17] и стал дальше думать об этом, мысли захватывали меня все сильнее.

Я даже подумывал написать свою собственную объектно-ориентированную методологию. К счастью, я скоро прекратил эти попытки, поскольку мне в руки попали труды Кента Бека по экстремальному программированию (XP).

Чем больше я читал об экстремальном программировании, тем больше я увлекался им. Идеи были революционны (по крайней мере, я тогда так думал). Они казались разумными, особенно в контексте объектно-ориентированного мышления (опять же на тот момент я думал именно так). Мне не терпелось узнать больше.

К моему удивлению, на той самой конференции в Мюнхене, посвященной объектно-ориентированному программированию, я заметил, что через зал от меня читает лекцию сам Кент Бек. Как-то раз во время перерыва я натолкнулся на него и предложил встретиться как-нибудь за завтраком, чтобы обсудить экстремальное программирование. На том завтраке был заложен фундамент для плодотворного партнерства. Наши обсуждения побудили меня полететь к нему в Медфорд, штат Орегон, чтобы совместно разрабатывать курс по экстремальному программированию.

В ходе этого визита я впервые попробовал поучаствовать в разработке через тестирование, и это меня увлекло.

В то время под моим управлением была компания Object Mentor. В сотрудничестве с Кентом Беком мы хотели предложить пятидневный учебный курс по экстремальному программированию, который назывался XP Immersion. С конца 1999-го по 11 сентября 2001 года[18] он производил настоящий фурор! Мы обучили сотни человек.

Летом 2000 года Кент Бек созвал кворум из сообщества по экстремальному программированию и паттернам. Встреча проходила недалеко от его дома. Он назвал ее встречей ведущих специалистов в области экстремального программирования. Мы катались на лодках и прогуливались по берегу реки Рог. И заодно решали, что делать дальше с экстремальным программированием.

Была идея создать некоммерческую организацию. Я ее горячо продвигал, но многие не разделяли моего энтузиазма. Видимо, у них был неблагоприятный опыт со схожей организацией в поддержку паттернов проектирования. Я был расстроен тем, как прошло заседание. Но Мартин Фаулер поддержал меня и предложил встретиться позже в Чикаго, чтобы все обсудить и выговориться. Я согласился.

С Мартином мы встретились осенью 2000 года в кафе неподалеку от офиса Thought Works, где он работал. Я описал ему свою идею собрать сторонников всех конкурирующих легковесных методологий и составить манифест, провозглашающий единство. Мартин сделал несколько рекомендаций касательно пригласительного списка. Мы вместе начали составлять приглашение. В тот же день, немного позже, я отправил это письмо. Темой письма было проведение встречи по обсуждению легковесных методологий.

---

[17] Это одно из тех странных совпадений, которые происходят время от времени. Нет ничего такого особенного в объектно-ориентированном программировании, что не дает применять в нем каскадную модель, тем не менее эта идея набирала в те дни популярность.

[18] Этот день очень значим, поэтому его стоило упомянуть.

Одним из приглашенных был Алистер Кокберн. Он позвонил мне и сказал, что тоже подумывал провести подобную встречу, однако наш список ему понравился больше, чем его собственный. Он предложил объединить наши пригласительные списки и договориться о встрече, если мы согласимся провести ее на горнолыжном курорте Сноуберд неподалеку от Солт-Лейк-Сити.

Итак, встреча намечалась в Сноуберде.

Сноуберд

Я был немало удивлен тем, что так много людей решило посетить мероприятие. Неужели кому-то действительно была интересна встреча, темой которой были легковесные методологии?

Однако мы все собрались в Сноуберде в гостиничном номере с прекрасным видом из окна.

Пришли 17 человек. С тех пор нас не раз критиковали за то, что все собравшиеся были белыми мужчинами среднего возраста. Критика была бы вполне справедлива, если бы не одно «но». Дело в том, что в списке приглашенных фигурировала одна женщина – Агнета Якобсон, но она не смогла приехать.

И, в конце концов, в то время во всем мире подавляющее большинство квалифицированных программистов было белыми мужчинами среднего возраста. А вот почему так сложилось – это отдельная история для совершенно другой книги.

У всех 17 из нас были довольно разные взгляды на пять различных легковесных методологий. Сторонников экстремального программирования было больше всех. Это были Кент Бек, Джеймс Греннинг, Уорд Каннингем, Рон Джеффрис и я.

Сторонников Scrum было немного меньше – Кен Швабер, Майк Бидл и Джефф Сазерленд.

Джон Керн высказывался в поддержку разработки, управляемой функциональностью, а Ариван Беннекум был сторонником метода разработки динамических систем. Наконец, Алистер Кокберн выступал за семейство методик, являвшихся его собственной разработкой – Crystal.

Остальные участники были относительно самостоятельны. Например, Энди Хант и Дейв Томас пропагандировали прагматизм в программировании. Они даже написали работу на эту тему. Брайан Марик был консультантом по тестированию. Джим Хайсмит был консультантом по управлению разработкой и сопровождению программного обеспечения. Стив Меллор следил за честностью каждого, потому что был сторонником подходов, управляемых моделями, к которым многие из нас относились с недоверием. И, наконец, присутствовал Мартин Фаулер. У него были личные взаимоотношения с командой, занимавшейся экстремальным программированием, однако к каким-либо «фирменным» методикам он относился довольно скептически. Мнения всех присутствующих он воспринимал благожелательно.

Я почти ничего не помню из того, что произошло за два дня нашей встречи. Другие участники событий видят картину по-своему, не так, как я[19]. Поэтому просто расскажу вам то, что сам помню. Расценивайте мои слова как воспоминания пожилого человека. Мне уже 65, а с того времени прошло почти два десятка лет. Возможно, я упустил несколько подробностей, но суть, думаю, передал правильно.

Каким-то образом мы решили, что я буду открывать встречу. Я поблагодарил всех за присутствие и высказал мнение, что наша цель состоит в составлении манифеста, в котором бы говорилось о разработке программного обеспечения в целом и описывались общие черты всех легковесных методологий, подмеченные нами. Закончив, я сел.

Я считаю, что это был мой единственный вклад в проведение встречи.

---

[19] Не так давно увидела свет история события, изложенная в литературном журнале The Atlantic за авторством Кэролайн Мимбс Найс: Caroline Mimbs Nyce. The winter getaway that turned the software world upside down // The Atlantic. 08.12.2017. URL: https://www.theatlantic.com/technology/archive/2017/12/agile-manifesto-a-history/547715/. Когда я это все писал, то еще не ознакомился с той статьей, поскольку мне не хотелось путать свои воспоминания, которые я изложил здесь.

Мы не занимались ничем необычным, когда записывали различные проблемы на карточках, а затем сортировали их на полу в группы по сходству. На самом деле я понятия не имею, что это нам дало. Просто помню, что мы это делали.

Затрудняюсь сказать, на какой день произошло чудо, – на первый или второй. Как мне кажется, это произошло к концу первого дня.

Возможно, именно группирование по сходству помогло нам выделить четыре ценности: личности и взаимодействие, рабочее программное обеспечение, взаимодействие с клиентами и реагирование на изменения. Кто-то написал это на магнитно-маркерной доске, находившейся в передней части комнаты. Затем ему в голову пришла блестящая мысль о том, что эти ценности приоритетны, но не заменяют остальные взаимодополняющие ценности методов, инструментов, документации, договоров и планов.

Это ключевая идея Манифеста Agile, и, кажется, никто отчетливо не помнит, кто первый обозначил ее на доске. Как мне помнится, это был Уорд Каннингем. Но сам Уорд приписывает это авторство Мартину Фаулеру.

Посмотрите на фотографию на сайте agilemanifesto.org. Уорд говорит, что сделал снимок, чтобы запечатлеть тот самый момент. На фото можно разглядеть Мартина Фаулера у доски и прочих участников встречи, которые собрались вокруг него[20]

[20] Слева направо, полукругом около Мартина Фаулера, на фотографии представлены: Дейв Томас, Энди Хант (или, возможно, Джон Керн), я (меня можно узнать по синим джинсам и мультитулу на ремне), Джим Хайсмит, кто-то, Рон Джеффрис и Джеймс Греннинг. Кто-то, уже не помню кто, сидел позади Рона. Возле его ботинка на полу, похоже, находится одна из карточек, которые мы использовали при группировке по сходству.

Конец ознакомительного фрагмента.