

Канбан

**АЛЬТЕР-
НАТИВНЫЙ
ПУТЬ
В AGILE**

ДЭВИД АНДЕРСОН

[Почитать описание, рецензии и купить на сайте МИФа](#)

СОДЕРЖАНИЕ

Предисловие	14
-------------------	----

Часть I. Основы

Глава 1. Решение дилеммы agile-менеджера	18
В поисках оптимального темпа	19
В поисках успешного управления изменениями	21
От системы «барабан-буфер-канат» к канбану	24
Возникновение Канбан-метода	26
Принятие Канбана в сообществе	26
Ценность Канбана неочевидна	28
Глава 2. Что такое Канбан-метод	30
Что такое канбан-система?	32
Применение канбана в разработке ПО	33
Зачем использовать канбан-систему	34
Канбан как комплексная адаптивная система для бережливого производства	36
Ситуационное поведение и Канбан	36
Канбан как разрешение действовать	37

Часть II. Преимущества Канбана

Глава 3. Рецепт успеха	42
Внедрение рецепта	43
Концентрация на качестве	45
Снижайте количество незавершенных задач и делайте частые релизы	48
Незавершенные задания (WIP), время выполнения и ошибки	48
Кто лучше?	52
Частые релизы порождают доверие	54
Неявное знание	55
Баланс между нагрузкой и пропускной способностью	56
Создание резерва времени	56
Расстановка приоритетов	57
Влияние	58

Рост зрелости	59
Атака на источники вариативности для улучшения предсказуемости	59
Рецепт успеха и Канбан	60
Глава 4. От худшего к лучшему за пять кварталов	62
Проблема	63
Визуализация рабочего процесса	64
Факторы, влияющие на производительность	65
Установление явных процедурных правил	67
Оценка была пустой тратой времени	67
Ограничение задач в работе (WIP)	68
Установление каденции пополнения	69
Достижение нового соглашения	70
Внедрение изменений	71
Адаптация правил	72
Поиск дальнейших улучшений	73
Результаты	75
Глава 5. Культура постоянного совершенствования	78
Культура кайдзен	79
Канбан повышает зрелость и возможности организации	80
Социологические изменения	87
Вирусное распространение сотрудничества	88
Культурные перемены — едва ли не главное преимущество Канбана	91
 Часть III. Внедрение Канбана	
Глава 6. Визуализация цепочки создания ценности	96
Определение стартовой и финишной контрольных точек	97
Типы работ	97
Создание стены карточек	99
Анализ нагрузки	103
Распределение мощности в соответствии с нагрузкой	104
Анатомия карточки	106
Системы управления задачами	108
Определение границ входа и выхода	109
Работа с параллельными процессами	110
Обработка неупорядоченной деятельности	112

Глава 7. Координация в канбан-системах	115
Визуальный контроль и вытягивание	115
Системы управления задачами	118
Ежедневные стендапы	119
Постсовещание	121
Собрания по пополнению очереди	121
Совещания по планированию поставок	123
Триаж	124
Анализ журнала проблем и эскалация наверх	126
Стикерные представители	128
Синхронизация по географическим зонам	128
Глава 8. Формирование каденции поставки	131
Координационные затраты на релиз	134
Операционные расходы релиза	135
Эффективность поставки	138
Формирование каденции поставки	139
Увеличение эффективности для ускорения каденции поставок	140
Релизы по запросу и по ситуации	141
Глава 9. Формирование каденции пополнения	146
Координационные расходы на расстановку приоритетов	146
Формирование каденции расстановки приоритетов	149
Эффективность расстановки приоритетов	150
Операционные расходы на расстановку приоритетов	151
Увеличение эффективности для сокращения каденции расстановки приоритетов	152
Расстановка приоритетов по мере необходимости или по запросу	154
Глава 10. Задание WIP-лимитов	158
Лимиты на рабочие задания	158
Лимиты на очереди	160
Буфер для бутылочного горлышка	161
Размер входящей очереди	162
Неограниченные разделы рабочего потока	164
Не подвергайте организацию стрессу	166
Не устанавливать WIP-лимит — это ошибка	167
Распределение мощности	168
Глава 11. Формирование соглашений об уровне обслуживания	171
Типичные определения классов обслуживания	172
Ускоренный класс обслуживания	173
Класс обслуживания «фиксированная дата поставки»	175

Стандартный класс	177
Нематериальный класс	178
Правила для классов обслуживания	180
Правила для ускоренного класса обслуживания	180
Правила для класса обслуживания с фиксированной датой поставки	181
Правила для стандартного класса обслуживания	182
Нематериальный класс обслуживания	183
Соглашение об уровне обслуживания	184
Назначение класса обслуживания	186
Использование классов обслуживания	187
Распределение мощности по классам обслуживания	188
Глава 12. Показатели и доклады для руководства	192
Отслеживание WIP	193
Время выполнения	194
Доля заданий, выполненных в срок	196
Пропускная способность	197
Проблемы и блокированные рабочие элементы	198
Эффективность потока	199
Первоначальное качество	200
Критическая нагрузка	201
Глава 13. Масштабирование Канбана	203
Иерархические требования	205
Разделение поставки ценности и вариативности рабочих единиц	206
Двухуровневые стены карточек	209
Введение «плавательных дорожек»	211
Альтернативный подход к борьбе с вариативностью трудозатрат	212
Введение классов обслуживания	212
Системная интеграция	213
Управление общими ресурсами	214
Глава 14. Операционный обзор	217
До совещания	217
Сразу задайте деловой тон	218
Приглашение гостей расширяет аудиторию и создает дополнительную ценность	219
Основная повестка	220
Ключ для перехода к бережливым принципам	221

Подходящая каденция	222
Демонстрация ценности менеджеров	224
Фокус на организации содействует кайдзену	224
Более ранний пример операционных обзоров	224
Глава 15. Начало перехода на Канбан	228
Культурные изменения, а не инициатива сверху	228
Основная цель для нашей канбан-системы	230
Цель 1. Оптимизация существующих процессов	230
Вторичные цели нашей канбан-системы	230
Цель 2. Высококачественные релизы	230
Цель 3. Повышение предсказуемости времени выполнения	231
Цель 4. Повышение удовлетворенности сотрудников	231
Цель 5. Создание резервов для дальнейшего совершенствования	232
Цель 6. Упрощение расстановки приоритетов	233
Цель 7. Обеспечение прозрачности дизайна и работы системы	235
Цель 8. Создание процесса, способствующего возникновению организации высокой степени зрелости	236
Знайте цели и формулируйте преимущества	237
Шаги для начала действий	238
Канбан предполагает иной тип сделки	241
Переговоры по внедрению Канбана	244
WIP-лимиты	245
Расстановка приоритетов	247
Релиз	247
Время выполнения и классы обслуживания	249

Часть IV. Совершенствование

Глава 16. Три типа возможностей для совершенствования	256
Бутылочные горлышки, устранение потерь и снижение вариативности	257
Теория ограничения	257
Пять направляющих шагов	259
Бережливое управление, TPS и устранение потерь	260
Система Деминга и «шесть сигм»	261
Совмещение Канбана с культурой вашей компании	264

Глава 17. Бутылочные горлышки и ограниченная доступность	266
Ресурсы ограниченной мощности	268
Увеличение мощности	269
Загрузка и защита	270
Подчинение ограничению	275
Ресурсы с ограниченной доступностью	276
Загрузка и защита	280
Подчинение ограничению	281
Увеличение мощности	282
Глава 18. Экономическая модель бережливого производства	287
Переосмысление «потерь»	287
Операционные расходы	288
Координационные расходы	291
Как узнать, что та или иная деятельность влечет за собой расходы	293
«Ошибочная» нагрузка	294
Глава 19. Источники вариативности	298
Внутренние источники вариативности	301
Размер единицы работы	301
Смещение типов единиц работы	302
Смещение классов обслуживания	304
Нерегулярный поток	305
Переработка	306
Внешние источники вариативности	307
Двойственность требований	307
Ускоренные запросы	309
Нерегулярный поток	311
Доступ к среде	313
Другие рыночные факторы	314
Трудности с координацией	315
Глава 20. Управление проблемами и правила эскалации	319
Управление проблемами	320
Эскалация проблем	322
Учет проблем и отчетность по ним	323
Примечания	327
Благодарности	329
Об авторе	332

ГЛАВА 1

РЕШЕНИЕ ДИЛЕММЫ AGILE-МЕНЕДЖЕРА

В 2002 году я был менеджером по разработке в удаленном офисе подразделения мобильных телефонов Motorola в Сиэтле (оно называлось PCS) и оказался в сложной ситуации. Мой отдел был частью стартапа, который Motorola приобрела годом ранее. Мы разрабатывали сетевое ПО для беспроводной передачи данных, например беспроводного скачивания и управления приборами. Эти серверные приложения входили в интегрированные системы, которые были тесно связаны с клиентским кодом мобильных телефонов, а также с другими элементами в телекоммуникационных сетях и операционной инфраструктуре, например с биллингом. Дедлайны назначались менеджерами, которые не обращали внимания на инженерную сложность проекта, его риски или масштаб. Основа кода развивалась из стартапа, при этом разработка многих первоначально запланированных возможностей была отложена на потом. Один старший разработчик настаивал на том, чтобы наши продукты назывались «прототипами». Нам было отчаянно необходимо повысить производительность и качество продукции, чтобы соответствовать требованиям бизнеса.

В своей повседневной деятельности в 2002 году и в процессе работы над предыдущей книгой¹ я был обеспокоен в основном двумя проблемами. Во-первых, как защитить команду от постоянно растущих требований бизнеса и достичь того, что сейчас в agile-сообществе принято

называть «оптимальным темпом». И во-вторых, как я могу успешно внедрить agile-подход в масштабах всей организации, преодолев неизбежное сопротивление переменам?

В ПОИСКАХ ОПТИМАЛЬНОГО ТЕМПА

В 2002 году agile-сообщество воспринимало оптимальный темп просто как «40-часовую рабочую неделю»². Принципы Agile-манифеста³ гласили, что «agile-процессы способствуют оптимальному развитию. Спонсоры, разработчики и пользователи должны быть готовы поддерживать постоянный темп в течение бесконечного времени». За два года до этого моя команда в Sprint PCS постоянно слышала от меня, что «масштабная разработка ПО — это марафон, а не спринт». Если членам команды предстояло поддерживать неизменный темп в процессе работы над полуторагодовым проектом, то нельзя было позволить им сгореть за месяц-другой. Проект нужно было распланировать, вставить в бюджет, расписать по времени и подвергнуть оценке, чтобы члены команды ежедневно тратили на работу разумное количество времени и не слишком уставали. Передо мной как менеджером стояла задача достичь этой цели и удовлетворить все требования бизнеса.

Когда я работал на первой менеджерской должности (это было в 1991 году, в стартапе, который делал платы захвата видео для персональных и более мелких компьютеров), CEO* сообщил, что у руководства сложилось обо мне «крайне отрицательное мнение». Я всегда отвечал «нет», когда наши возможности как разработчиков уже были исчерпаны, а от нас требовали все больше продуктов или функций. К 2002 году это вошло у меня в привычку: я провел еще десять лет, отказываясь выполнять капризы владельцев бизнеса.

* Chief Executive Officer — главный исполнительный директор (генеральный директор).
Прим. ред.

Команды разработчиков и IT-отделы компаний сильно зависят от других групп, которые постоянно торгуются, упраснивают, угрожают и переделывают даже самые очевидные и объективно разработанные планы. В число уязвимых попадают и планы, основанные на тщательном анализе и историческом опыте. Большинство же команд, не имевших тщательных методов анализа и исторического опыта, не могли совладать с теми, кто подталкивал их брать на себя непонятные, а нередко и неосуществимые обязательства.

Тем временем сотрудники смирились с безумной загрузкой, и непомерные объемы работы стали нормой. Кажется, никто не задумывался над тем, что у инженеров-программистов тоже может быть общественная или семейная жизнь. Звучит грубо, но это правда! Я знаю слишком много примеров, когда излишняя производственная нагрузка навсегда разрушила семейные отношения. Трудно сочувствовать типичному гик-разработчику. В моем родном штате Вашингтон доход инженера-программиста уступает только заработку стоматолога. Как и во времена Форда, то есть в 1920-е годы, когда люди на его заводах зарабатывали в пять раз больше, чем в среднем по стране, никому и в голову не приходит подумать о монотонности работы или о благополучии специалистов: им столько платят! Трудно представить себе наличие профсоюзов в таких интеллектуальных отраслях, как разработка ПО, потому что никто не станет всерьез изучать причины физических и психологических недугов, которые испытывают разработчики. Более ответственные работодатели предлагают, например, такие меры, как массаж или психотерапия. Или проводят дни психического здоровья — и это вместо того, чтобы уделить внимание изучению основных причин проблемы. Технический писатель, сотрудник известной фирмы — разработчика ПО, однажды сказал мне: «Нет ничего страшного в том, что я употребляю антидепрессанты, ведь так поступают все!» Программисты обычно соглашались со всеми требованиями, получают неплохую зарплату и страдают от последствий. Я хотел изменить такое положение дел, найти взаимовыгодный

подход, который позволял бы мне говорить «да» и при этом все же защищать свою команду, обеспечивая достижение оптимального темпа. Я хотел вернуть членов своей команды в общество и семью и улучшить условия, которые вызывали у разработчиков, не достигших и тридцати лет, стресс и проблемы со здоровьем. Поэтому я решил начать бороться с этими проблемами.

В ПОИСКАХ УСПЕШНОГО УПРАВЛЕНИЯ ИЗМЕНЕНИЯМИ

Вторая проблема, которая занимала мои мысли, — это управление изменениями в крупных организациях. Я был менеджером по разработке в Sprint PCS и затем в Motorola. В обеих компаниях существовала серьезная потребность в переходе на более гибкие методы работы. Но в обоих случаях у меня возникали трудности при внедрении agile-методов более чем в одной-двух командах.

Оба раза у меня не было достаточно полномочий, чтобы просто приказывать внести изменения в работу множества команд. Я старался внедрить изменения по просьбе высшего руководства, но не обладал должной властью. Меня просили повлиять на коллег, чтобы те внедрили в своих командах такие же изменения, как я — в своей. Но они не торопились брать на вооружение методы, которые, казалось бы, зарекомендовали себя в моей команде наилучшим образом. У этого сопротивления, вероятно, было несколько причин. Чаще всего я слышал, что у каждой команды своя ситуация и мои методы нужно будет подгонять под конкретные нужды других. К середине 2002 года я понял, что жестко предписывать какой-либо процесс разработки ПО бесполезно — это просто не будет работать.

Процесс нужно было адаптировать для каждой конкретной ситуации, поэтому требовалось активное руководство каждой командой. А этого нередко не хватало. Даже при должном руководстве нет полной уверенности

в том, что существенные изменения могут произойти без наличия установленной структуры и советов по поводу того, как подогнать процессы под иные ситуации. Если у руководителя, коуча или ответственного инженера нет четкого представления о том, что делать, то любая адаптация, скорее всего, пройдет субъективно. При этом велика вероятность ошибок — например, внедрения неподходящего шаблона процесса.

Я попытался осветить эту проблему в книге *Agile Management for Software Engineering*, которую писал в то время. Я задавался вопросом: «Почему гибкая разработка дает лучшие экономические результаты, чем традиционные подходы?» Я хотел применить с этой целью структуру теории ограничений⁴.

В процессе исследований и написания упомянутой книги я понял, что уникальна каждая ситуация. Да и разве может сдерживающий фактор или узкое место оказаться одинаковым для любой команды и проекта в любое время? Каждая команда уникальна: иные навыки, возможности, опыт. Каждый проект отличается от других бюджетом, расписанием, объемом и рисками. Непохожи друг на друга и организации: у них разные цепочки создания ценности, они работают на различных рынках (рис. 1.1). Мне показалось, что это может дать ключ к пониманию сопротивления изменениям. Если предлагаемые перемены в методах работы и моделях поведения не дают, по мнению сотрудника, очевидного преимущества, то он не примет их. Если эти изменения не влияют на то, что воспринимается командой как ограничитель или сдерживающий фактор, то команда будет сопротивляться. Иными словами, перемены, предложенные без учета контекста, будут отвергнуты сотрудниками, которые прекрасно знают контекст работы.

Казалось бы, будет лучше, если новый процесс начнет развиваться, устраняя одно ограничение за другим. Это основное положение теории ограничений Голдратта. Понимая, что мне еще многому предстоит научиться, я осознавал ценность материала и устремился вперед в работе над рукописью. Мне было ясно, что книга не давала советов, как внедрить

идеи в более широком масштабе, а также почти не помогала найти способы управления изменениями.

У команд разные: навыки, опыт, возможности	У проектов разные: бюджеты, расписания, объемы, риски
У организаций разные: цепочки создания ценности, целевые рынки	

Рис. 1.1. Почему универсальные методологии разработки неверны

Подход Голдратта, изложенный в главе 16, направлен на поиск ограничений, а затем и способов их устранения, чтобы они перестали препятствовать производительности. После этого возникает новое ограничение, и цикл повторяется. Это итеративный подход, предполагающий систематическое улучшение производительности посредством выявления и устранения препятствий.

Я понял, что можно сочетать этот подход с некоторыми приемами из области бережливого производства. Смоделировав рабочий процесс жизненного цикла разработки ПО как потока создания ценности и создав систему отслеживания и визуализации для фиксации изменений состояния возникающей работы, «протекающей» по системе, я мог определить ограничители. Способность выявить ограничение — это первый шаг к модели, лежащей в основе ТОС. Голдратт уже разработал применение этой теории для проблем рабочего процесса, носящее неуклюжее название «барабан-буфер-канат». Однако я понял, что упрощенный вариант этой системы можно внедрить в область разработки ПО.

С точки зрения происхождения «барабан-буфер-канат» — это пример класса решений, известных как вытягивающие системы. Как мы увидим

в главе 2, канбан тоже один из примеров такого рода систем. Побочный эффект вытягивающих систем состоит в том, что они ограничивают объем незавершенной работы до установленного заранее количества, препятствуя перегрузке сотрудников. К тому же полностью загруженными остаются только работники, напрямую сталкивающиеся с ограничением; у всех остальных должно оставаться свободное время. Я понял, что вытягивающие системы способны решить обе волновавшие меня проблемы. Вытягивающая система позволит мне внедрять пошаговые изменения, что (как я надеялся) существенно уменьшит сопротивление им, а также облегчит достижение оптимального темпа. Я принял решение перейти на систему «барабан-буфер-канат» при первой возможности. Мне хотелось поэкспериментировать с пошаговой эволюцией процесса и посмотреть, насколько она обеспечивает оптимальный темп и снижает сопротивление изменениям.

Такая возможность появилась осенью 2004 года в Microsoft, что подробно описано в главе 4 этой книги в анализе примера.

ОТ СИСТЕМЫ «БАРАБАН-БУФЕР-КАНАТ» К КАНБАЛУ

Применение решения «барабан-буфер-канат» в Microsoft дало свои результаты. Сопротивление было слабым, производительность выросла более чем втрое, время опережения сократилось на 90%, а предсказуемость повысилась на 98%. Осенью 2005 года я сообщил о достигнутых результатах на конференции в Барселоне⁵, а зимой 2006 года сделал еще один доклад. Моя работа привлекла внимание Дональда Рейнертсена, который специально приехал ко мне в офис в Редмонде. Он хотел убедить меня, что все готово к полному переходу на канбан.

Кан-бан — японское слово, которое дословно переводится как «сигнальная доска». В производстве такая доска используется для визуализации нарастающего темпа производства, что позволяет давать больше

продукции. Сотрудники на каждом этапе процесса не могут перейти к следующей фазе работы, пока посредством канбан-доски не будет дан соответствующий сигнал. Хотя я знал о существовании этого механизма, я не был убежден, что он полезен или даже жизнеспособен применительно к интеллектуальной работе, особенно к разработке ПО. Я понимал, что канбан обеспечивает оптимальный темп, но не знал о его хорошей репутации в качестве метода стимулирования поэтапного улучшения процессов. Я не знал, что Тайити Оно, один из создателей производственной системы Toyota, говорил: «Два основных принципа системы производства Toyota — это “точно-в-срок” и автоматизация с участием человека, или автономизация. Для управления системой используется инструмент — это и есть канбан». Иными словами, канбан жизненно важен для процесса *кайдзен* («непрерывное улучшение»), применяемого в Toyota. Это механизм, который заставляет систему работать. Сейчас, имея за плечами пятилетний опыт, я принимаю это как абсолютную истину.

К счастью, Дон привел убедительный аргумент в пользу перехода с системы «барабан-буфер-канат» на канбан. Он звучал довольно эзотерически: система канбан обеспечивает более гладкий переход с простоя в узком месте, чем «барабан-буфер-канат». Однако понимание подобной отличительной особенности не так уж обязательно, чтобы читать и понимать эту книгу.

Вновь изучив реализованное в Microsoft решение, я понял, что если бы мы сразу воспринимали его как результат системы канбан, то итог был бы таким же. Мне показалось интересным, что два разных подхода ведут к одному и тому же результату. Итак, поскольку получался один и тот же процесс, я не чувствовал себя обязанным воспринимать его исключительно как результат внедрения системы «барабан-буфер-канат».

Я стал предпочитать этому сложному словосочетанию термин «канбан». Он используется в бережливом производстве (то же, что производственная система Toyota). Эта совокупность знаний получила гораздо большее распространение и признание, чем теория ограничений.

Канбан, несмотря на свое японское происхождение, менее метафоричен, чем барабан, буфер и канат, вместе взятые. Это слово легче произносить, объяснять и, как оказалось впоследствии, преподавать и внедрять. Вот оно и закрепилось.

ВОЗНИКНОВЕНИЕ КАНБАН-МЕТОДА

В сентябре 2006 года я ушел из Microsoft и возглавил отдел разработки ПО в Corbis — расположенной в центре Сиэтла частной компании по хранению базы фотографий и защите интеллектуальной собственности. Вдохновившись достигнутым в Microsoft, я решил внедрить вытягивающую систему канбан в Corbis. И здесь результаты были весьма успешными, они привели к развитию большинства концепций, представленных в этой книге. Это расширенный набор тех концепций — визуализация рабочего процесса, типы элемента потока операций, каденции, классы обслуживания, особая отчетность руководства и анализ операций, — которые определяют Канбан-метод.

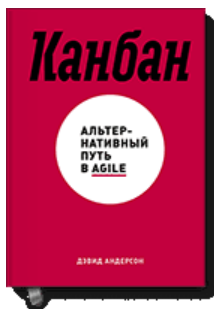
В этой книге я описываю Канбан (с большой буквы) как метод эволюционных изменений, использующий вытягивающую систему канбан (с маленькой буквы), визуализацию и другие инструменты для катализации введения идей бережливого производства в технологические разработки и IT-операции. Это эволюционный и пошаговый процесс. Канбан позволяет достичь оптимизации процесса, связанной с контекстом, с минимальным сопротивлением изменениям и при этом сохранить оптимальный темп для всех вовлеченных в работу сотрудников.

ПРИНЯТИЕ КАНБАНА В СООБЩЕСТВЕ

В мае 2007 года Рик Гарбер и я представили первые результаты работы в Corbis на конференции по бережливой разработке новых продуктов

в Чикаго. Доклад слушали примерно 55 человек. Летом того же года на конференции Agile 2007 в Вашингтоне я предложил провести круглый стол по обсуждению системы канбан — на него пришло человек двадцать пять. Через два дня один из присутствовавших, Арло Бельши, произнес краткую речь, в которой рассказывал о своем методе «обнаженного планирования»⁶. Оказалось, что и другие компании внедряют у себя вытягивающие системы. Так, в Yahoo! была создана дискуссионная группа, которая быстро набрала сотню членов. А сейчас в нее входит уже более 1000 человек. Некоторые из участников моего круглого стола решили попробовать Канбан на своем рабочем месте — нередко с командами, которые безуспешно боролись со Scrum. Самые известные из моих ранних последователей — Карл Скотланд, Аарон Сандерс и Джо Арнольд, все из Yahoo!. Эта компания быстро внедрила Канбан в более чем десяток команд на трех континентах. Еще один заметный участник круглого стола, Кендзи Хиранабе, разрабатывал канбан-решения в Японии. Вскоре после конференции он написал на эту тему две статьи для InfoQ^{7,8}, которые вызвали большой интерес. Осенью 2007 года Санджив Огастайн, автор Managing Agile Projects⁹ и основатель Организации руководителей гибких проектов (Agile Project Leadership Network, APLN), посетил Corbis в Сиэтле и описал наши канбан-системы как «первый новый agile-метод, который я увидел за пять лет».

На следующий год на конференции Agile 2008 в Торонто состоялось шесть презентаций решений канбан разного формата. Одна из них была проведена Джошуа Кериевски из Industrial Logic — компании по обучению и консалтингу в области экстремального программирования. В ней он продемонстрировал, как пришел к похожим идеям по взятию на вооружение принципов экстремального программирования и их улучшению в контексте его бизнеса. В тот год Agile Alliance вручил приз Гордона Паска Арло Бельши и Кендзи Хиранабе за их вклад в agile-сообщество. Как я понял, он состоял в одном случае во внедрении Канбана, а в другом — в разработке и пропаганде на удивление схожих идей.



Почитать описание и заказать
в МИФе

Смотреть книгу

Лучшие цитаты из книг, бесплатные главы и новинки:

Взрослые книги:



Проза:



Детские книги:

