Гибкие технологии управления проектами

Лектор – Скороход С. В.

Классический подход к управлению проектами - водопад

Классический подход к управлению проектами

Этап 1. Инициация.

- Руководитель проекта и команда определяют требования к проекту.
- На этом этапе проводятся совещания и «мозговые штурмы», на которых определяется что же должен представлять из себя продукт проекта.

Этап 2. Планирование.

- Команда решает, как она будет достигать цели, поставленной на предыдущем этапе.
- Команда уточняет и детализует цели и результаты проекта, а также состав работ по нему.
- На основании данной информации команда формирует календарный план и бюджет, оценивает риски и выявляет заинтересованные стороны.

Этап 3. Разработка.

- Выбор комплекса инструментальных средств, разработка математических методов и алгоритмов.

Классический подход к управлению проектами

Этап 4. Реализация и тестирование.

- Основная работа по проекту – написание кода, разработка устройства и т.п..
- Следуя разработанным планам начинает создаваться содержание проекта, определённое ранее, проводится контроль по выбранным метрикам.
- Во второй части данной фазы происходит тестирование продукта, он проверяется на соответствие требованиям Заказчика и заинтересованных сторон.
- В части тестирования выявляются и исправляются недостатки продукта.

Этап 5. Мониторинг и завершение проекта.

- В зависимости от проекта данная фаза может состоять из простой передачи Заказчику результатов проекта или же из длительного процесса взаимодействия с клиентами по улучшению проекта и повышению их удовлетворённости, и поддержке результатов проекта.

Классический подход к управлению проектами: достоинства

- Требует от Заказчика и руководства компании определить, что же они хотят получить, уже на первом этапе проекта.
- Раннее включение привносит стабильность в работу проекта.
- Планирование позволяет упорядочить реализацию проекта.
- Этот подход подразумевает мониторинг показателей и тестирование, что совершенно необходимо для реальных проектов различного масштаба.
- Классический подход позволяет избежать стрессов ввиду наличия запасного времени на каждом этапе, заложенного на случай каких-либо осложнений и реализации рисков.
- С правильно проведённым этапом планирования, руководитель проектов всегда знает, какими ресурсами он обладает. Даже если эта оценка не всегда точная.

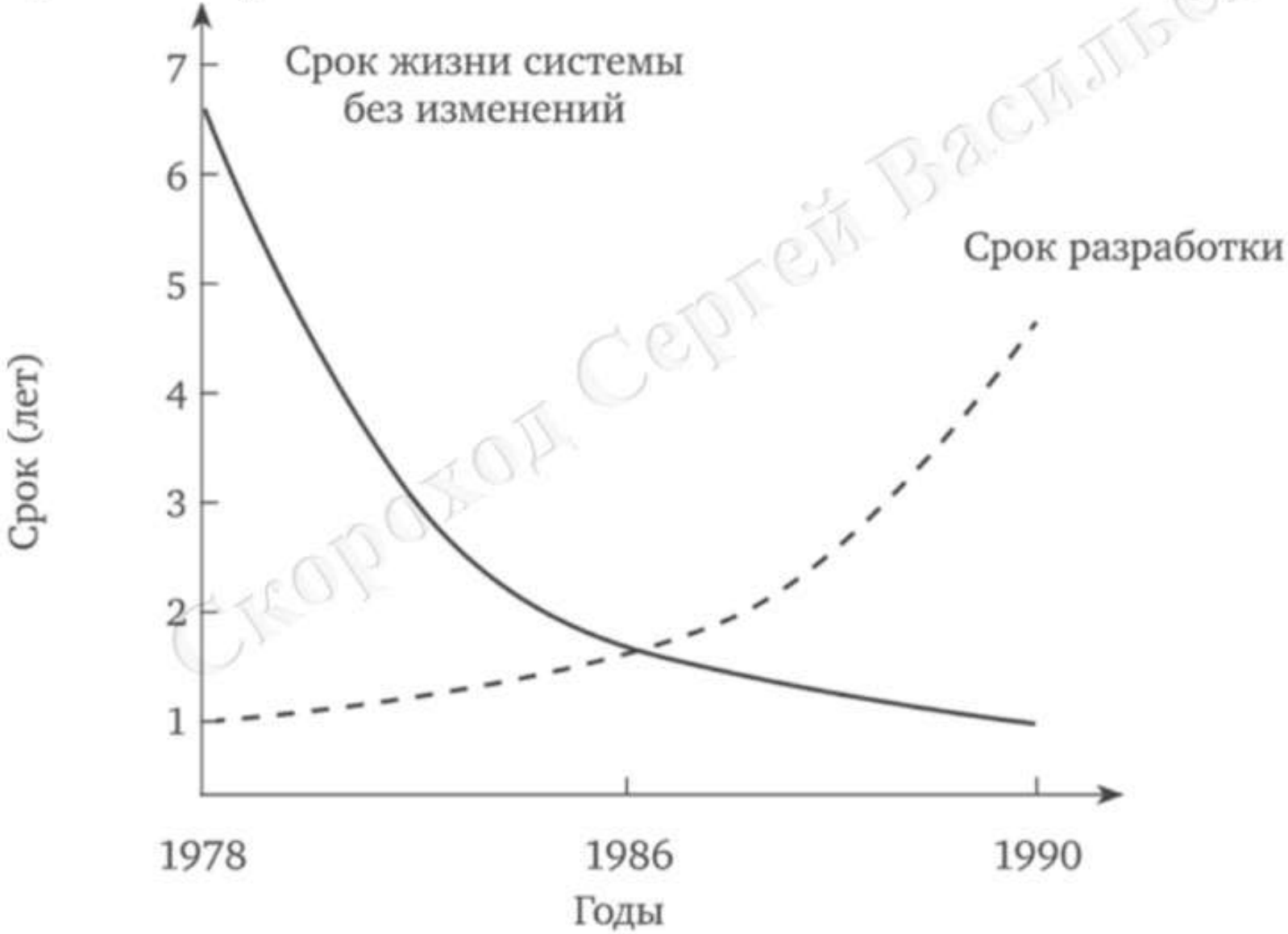
Классический подход к управлению проектами: недостатки

- Нетолерантность к изменениям.
- Если в Вашем проекте ресурсы и время не являются ключевыми ограничениями, а содержание проекта подвержено изменениям – возможно вам стоит присмотреться к гибким технологиям управления проектами.

Классический подход : недостатки

Предпосылки Agile

1. Противоречие между ростом времени на первичное создание автоматизированных систем и необходимостью их быстрого эволюционного развития



Предпосылки Agile

1. Противоречие между ростом времени на первичное создание автоматизированных систем и необходимостью их быстрого эволюционного развития

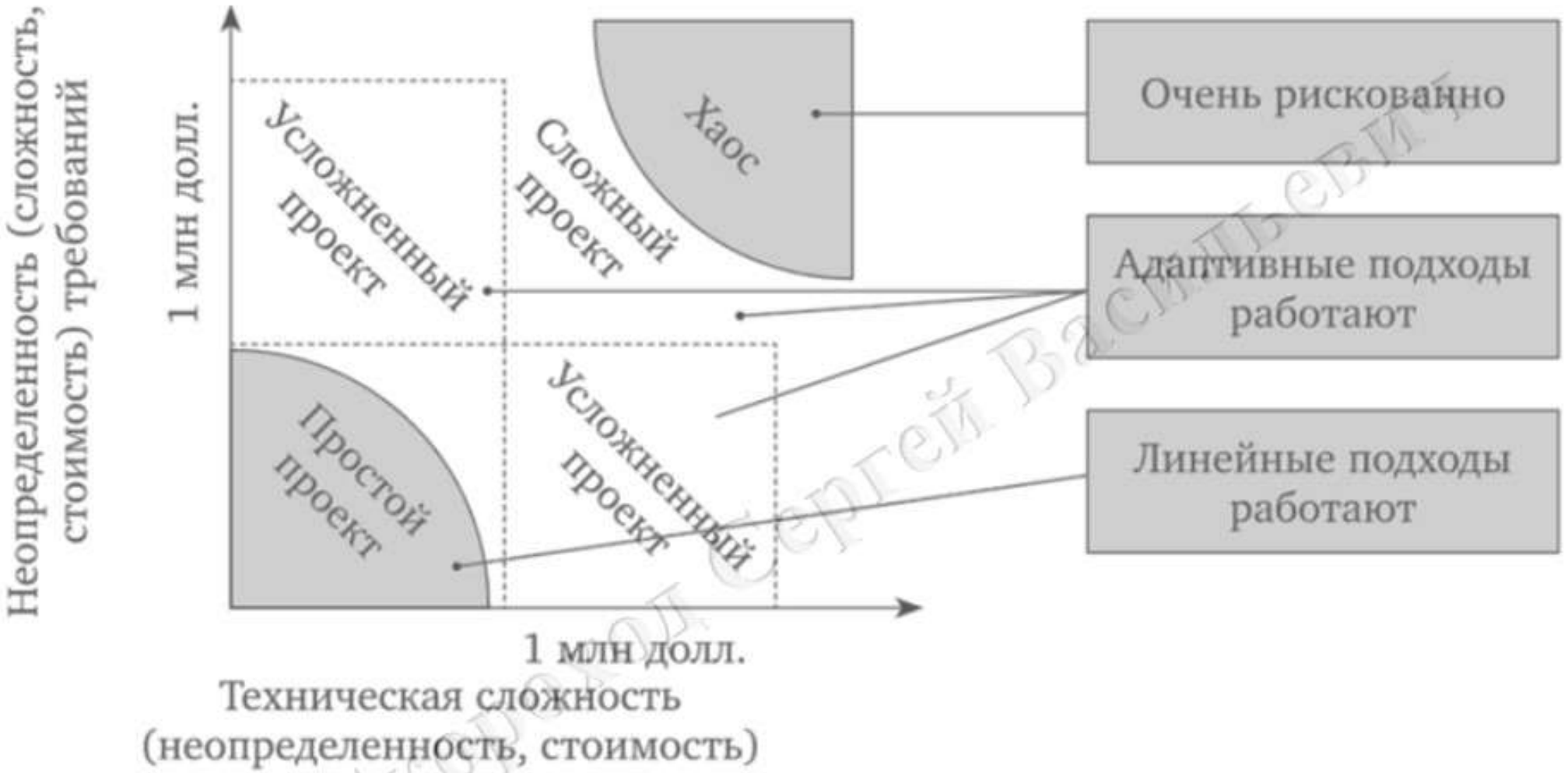
- 1970-е гг – срок разработки системы – 1 год, срок эксплуатации системы без необходимости изменений – 7 лет.
- 1986 г – сроки разработки и эксплуатации системы без изменений сравнялись. Непосредственно после внедрения новой системы начинался цикл ее обновления.
- 1990 гг – наше время - срок разработки промышленных прикладных систем начал превышать срок их жизни без изменений.
- Современную большую прикладную систему невозможно разработать и внедрить, пользуясь традиционным водопадным подходом — требования к системам в большинстве случаев начинают изменяться задолго до финиша проекта.

Предпосылки Agile

2. Непрерывное усложнение автоматизированных систем.

- Производительность компьютеров нарастает, количество обрабатываемых объектов постоянно растет (с темпом около 1000 раз за 15 лет).
- Автоматизированные системы объединяются в распределенные вычислительные комплексы, которые многократно усложняют анализ и проектирование взаимодействия компонент и при этом охватывают единовременно все большее количество процессов предприятия и даже групп предприятий.
- Характеристики и функционал новой создаваемой системы становятся непредсказуемыми не только в точке старта проекта, но и практически до самого его завершения — последние штрихи в характеристики продукта вносятся даже на этапе тестирования и опытной эксплуатации.
- С ускоряющимся темпом изменяется и внешняя для предприятия конкурентная среда, внося свой вклад в неопределенность продукта.

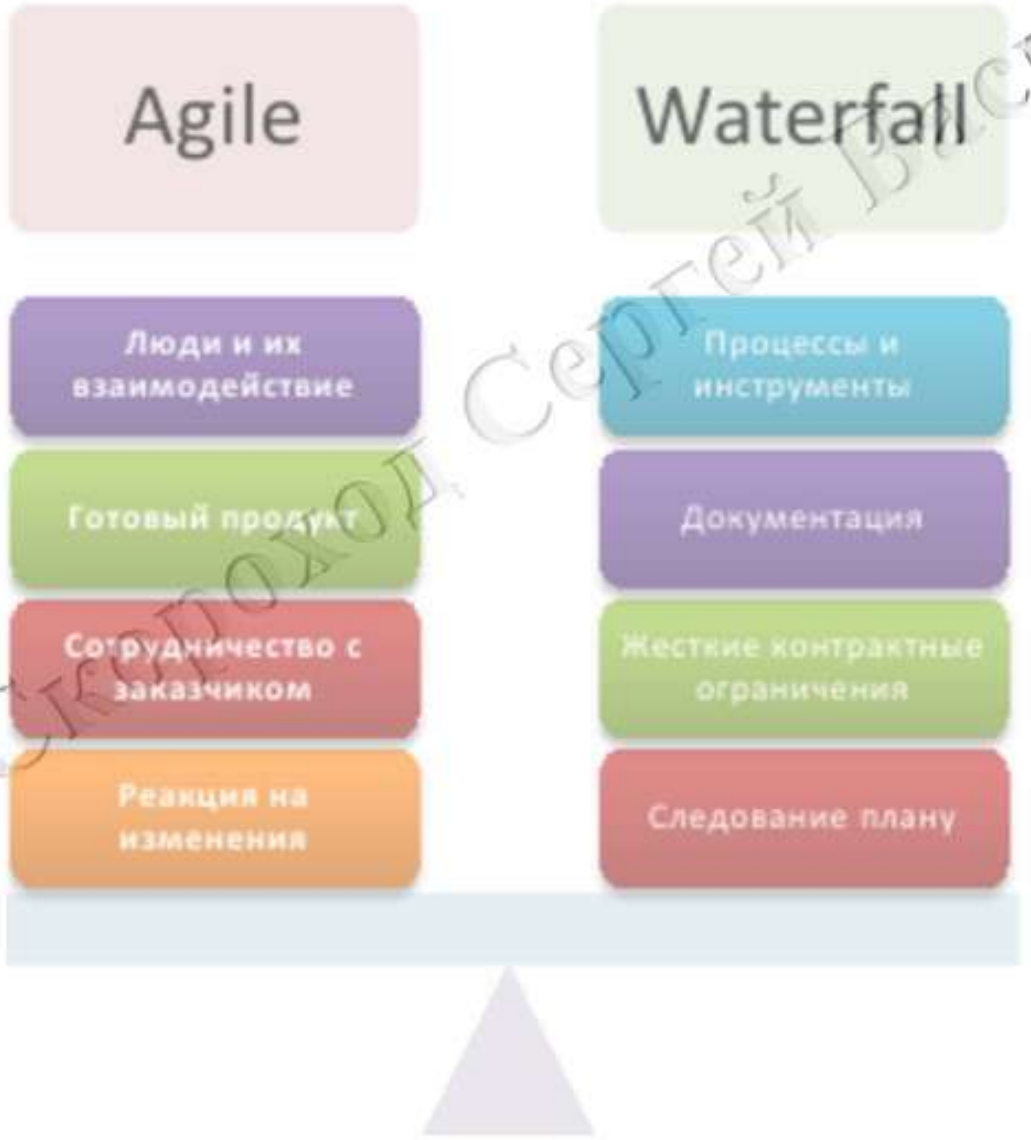
Предпосылки Agile



- Проекты стоимостью до 0,4 млн долл., реализуемые с подходом водопад, успешны в 44 % случаев.
- Проекты стоимостью от 1,3 млн долл., реализуемые с подходом водопад, успешны всего в 7 % случаев.

Манифест Agile

Люди и их взаимодействие важнее процессов и инструментов;
Готовый продукт важнее документации по нему;
Сотрудничество с заказчиком важнее жестких контрактных ограничений;
Реакция на изменения важнее следования плану.



Манифест Agile

1. Люди и их взаимодействие важнее процессов и инструментов.

Чтобы люди работали эффективнее, процессы и инструменты не должны их ограничивать. В Agile ни процесс, ни тем более программный инструмент не диктует, что людям делать. Более того, они сами решают, как менять процессы/инструменты своей работы.
Чтобы ускорить процесс разработки, люди также должны взаимодействовать напрямую (без посредников в виде документов или других людей), активно общаться между собой лично, а не письменно. Правда, в современном бизнесе общение часто вынуждено переходить в онлайн. Но тогда это должна быть видеосвязь с интерактивными онлайн-досками, а не только письма и чаты.

Манифест Agile

2. Работающий продукт важнее исчерпывающей документации.

Чтобы клиенты были довольны, им нужен именно работающий продукт. Поэтому разработчики продукта должны фокусироваться именно на том, чтобы продуктом можно было как можно скорее воспользоваться, а не на составлении списков, диаграмм, требований, отчетов перед заказчиком.

Чтобы укладываться в сжатые сроки с минимумом затрат, зачастую не стоит связывать себя документацией. Поддержка документации в адекватном продукту состоянии нередко замедляет разработку и требует неоправданно больших затрат.

Манифест Agile

3. Сотрудничество с заказчиком важнее согласований условий контракта.

Чтобы на выходе получить продукт, действительно ценный для заказчика, стоит отказаться от излишних деталей в контракте между подрядчиком и заказчиком (равно как и в требованиях внутреннего заказчика к внутреннему разработчику продукта). Будучи жестко заданы на старте, детали контракта мешают учитывать новые данные и приоритеты, появляющиеся лишь во время разработки.

Чтобы бизнес-ценность продукта быстро росла, заказчик с разработчиком должны плотно общаться по ходу работы. В этом случае все возникающие изменения и проблемы оперативно обрабатываются обеими сторонами.А чтобы такое сотрудничество исполнителя и заказчика стало возможным, нужно выстраивать их доверие друг к другу.

Манифест Agile

4. Готовность к изменениям важнее, чем следование плану.
Чтобы не откладывать риски проектов на последние стадии разработки (когда будет уже поздно уменьшать содержание работы, сдвигать срок или усиливать команду), Agile предлагает не только итеративность работы, но и готовность к изменениям на всех стадиях.
Чтобы в первую очередь делалось самое ценное, текущее видение бизнес-ценности и позиционирования продукта должно быть прозрачно для разработчиков, а процесс их работы должен позволять вносить существенные изменения в прежние планы. В том числе, разработчики должны быть готовы добавлять в продукт незапланированные новые возможности, если они стали ценными в изменившейся ситуации. Готовность заказчика оперативно жертвовать какой-то частью запланированного также нужна в ситуации, когда исполнители столкнулись с непредвиденными проблемами в ходе разработки.

Agile — философия

Agile — это не методология разработки, а система ценностей, помогающих разработчикам делать новые продукты быстрее и с бОльшим эффектом для бизнеса:

- за счет более эффективного взаимодействия с заказчиком и друг с другом, которое не ограничивается жестким контрактом или жестким внутренним процессом;
- за счет быстрой реакции на изменения, причем с обеих сторон;
- за счет фокуса на работающий продукт, а не на вспомогательные вещи вроде документации.

- Ценности эти настолько общие и даже абстрактные, что Agile часто называют философией.
- Образ мышления Agile реализуется через фреймворк Scrum, метод Kanban и другие техники.

Схема работы по Agile

- Проект разбивается не на последовательные фазы, а на маленькие подпроекты, которые затем «собираются» в готовый продукт.
- Инициация и верхнеуровневое планирование проводятся для всего проекта, а последующие этапы: разработка, тестирование и прочие проводятся для каждого мини-проекта отдельно.
- Это позволяет передавать результаты этих мини-проектов, так называемые, инкременты, быстрее, а приступая к новому подпроекту (итарации) в него можно внести изменения без больших затрат и влияния на остальные части проекта.

Сильные и слабые стороны Agile

Сильные стороны Agile

- Гибкость и адаптивность. Он может подстроиться под практически любые условия и процессы организации.
- Именно быстрая и относительно безболезненная реакция на изменения является причиной тому, что многие крупные компании стремятся сделать свои процессы более гибкими.
- Agile отлично подходит для проектов с «открытым концом» — например, запуску сервиса или блога.
- Вотчина Agile – разработка новых, инновационных продуктов. В проектах по разработке таких продуктов высока доля неопределённости, а информация о продукте раскрывается по ходу проекта.

Слабые стороны Agile

- Agile — это набор принципов и ценностей.
- Каждой команде придётся самостоятельно составлять свою систему управления, руководствуясь принципами Agile.
- Это непростой и длительный процесс, который потребует изменений всей организации, начиная процедурами и заканчивая базовыми ценностями. Это тернистый путь и не всем организациям он под силу.

Фреймворк Scrum

- Создан в 1986 году.
- Считается самым структурированным из семейства Agile.
- Сочетает в себе элементы классического процесса и идеи гибкого подхода к управлению проектами.
- В Scrum работа ведется спринтами — одинаковыми по продолжительности короткими итерациями. Вся работа выполняется силами небольшой (до 10 человек) команды, в которую входят разработчики, владелец продукта (отвечающий за успех продукта) и скрам-мастер (отвечающий за эффективность и правильное применение Scrum). Команда самостоятельно решает, кто, что, когда и как делает.

Элементы Scrum

Спринт

- Итерация разработки, в ходе которой создается новый результат.
- Жестко фиксирован во времени – от 1 до 4 недель.
- Чем короче спринт, тем гибче процесс разработки, поскольку после каждого спринта требования к системе могут корректироваться на основании обратной связи от заказчика.
- С другой стороны при более длительном спринте снижаются издержки на совещания, и больше остается времени на решение задач проекта.

Беклог продукта

- Журнал пожеланий к продукту.
- Это список требований к системе, упорядоченный по приоритету – важности реализации.
- Приоритет исчисляется в очках. Система очков устанавливается командой самостоятельно.
- Журнал пожеланий могут дополнять все участники процесса.

Элементы Scrum

Беклог спринта

- Журнал пожеланий для реализации в ходе спринта.
- Содержит функциональность, отобранную владельцем проекта для реализации на текущем спринте.

Встреча по упорядочиванию беклога продукта

- Эта встреча аналогична фазе планирования в классическом проектном управлении, и проводится в первый день каждого Спринта.
- На ней рассматривается – что уже было сделано по проекту в целом, что ещё осталось сделать и принимается решение о том, что же делать дальше.
- Владелец продукта определяет, какие задачи на данном этапе являются наиболее приоритетными.
- Данный процесс определяет эффективность Спринта, ведь именно от него зависит, какую ценность получит Заказчик по итогам спринта.

Элементы Scrum

Планирование Спринта

- После того, как Владелец продукта определил приоритеты, команда совместно решает, что же конкретно они будут делать во время грядущей итерации, как достигнуть поставленные перед спринтом цели.
- Команды могут применять различные инструменты планирования и оценки на данном этапе, лишь бы они не противоречили принципам и логике Scrum.
- Планирование Спринта проводится в самом начале итерации, после Встречи по упорядочиванию беклога продукта.

Ежедневные летучки

- проводится в одно и то же время, в одном и том же месте;
- не более 15 минут;
- каждому участнику надо ответить на 3 вопроса:
  - Что я сделал вчера.
  - Что я планирую сделать сегодня.
  - Что мне мешает.

Элементы Scrum

Подведение итогов Спринта (до 4-х часов)

- Проводится в конце спринта.
- Команда демонстрирует результаты спринта всем заинтересованным лицам. Привлекается максимальное количество зрителей.
- Все члены команды участвуют в демонстрации (один человек на демонстрацию или каждый показывает, что сделал за спринт).
- Нельзя демонстрировать незавершенную функциональность.

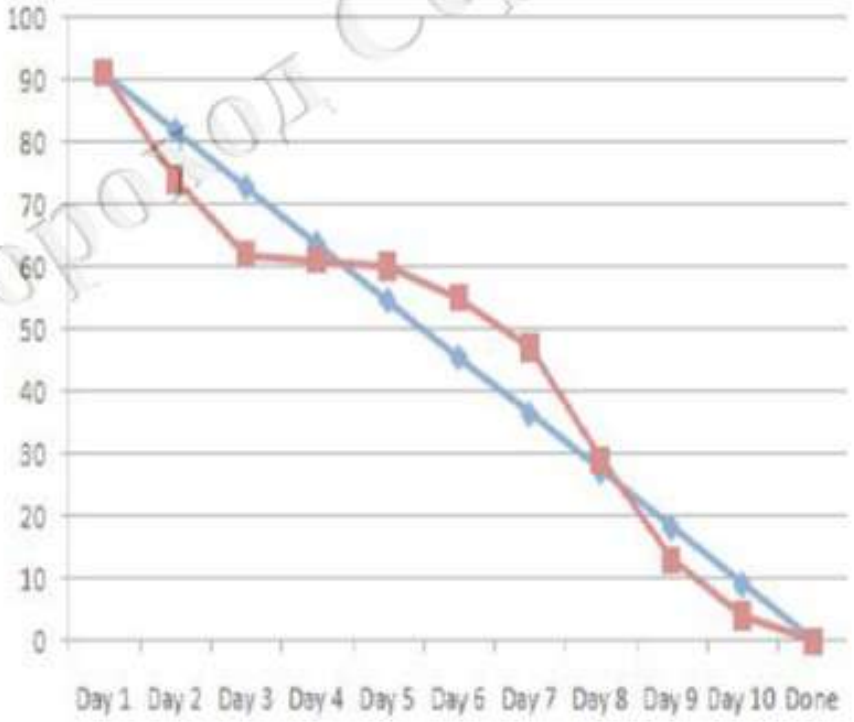
Ретроспектива Спринта (1-3 часа)

- Проводится в конце спринта. Обсуждение результатов спринта.
- Члены команды высказывают своё мнение о прошедшем спринте.
- Отвечают на два основных вопроса:
  - Что было сделано хорошо в прошедшем спринте?
  - Что надо улучшить в следующем?
- Выполняют улучшение процесса разработки (решают вопросы и фиксируют удачные решения).

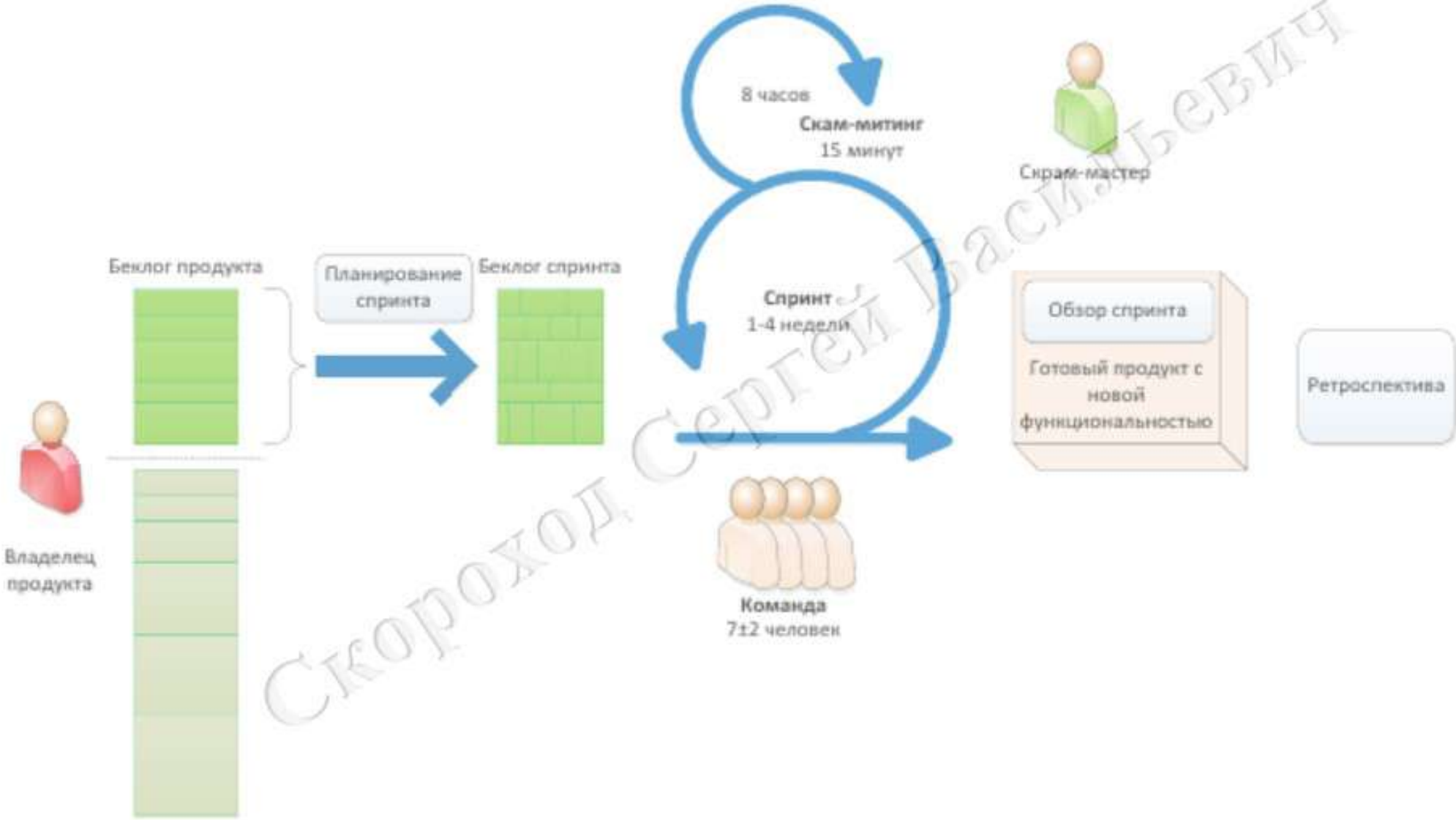
Элементы Scrum

Диаграмма сгорания задач

- Демонстрирует объем сделанной и оставшейся работы относительно срока проекта.
- Диаграмма актуализируется ежедневно.
- Предусмотрены два вида диаграмм:
  - Диаграмма сгорания для спринта — показывает, сколько уже задач сделано и сколько ещё остаётся сделать в текущем спринте.
  - Диаграмма сгорания для проекта — показывает, сколько уже задач сделано и сколько ещё остаётся сделать до завершения проекта.



Scrum-разработка



Роли в Scrum: Владелец продукта

- Это член команды, который отвечает за ценность проделываемой командой работы, несет ответственность перед заинтересованными лицами, является представителем заказчика.
- Обязанности: своевременное предоставление требований к продукту, определение дат и содержания релизов.
- Единственный человек в Команде, кто отвечает за создание и контроль беклога продукта, за расстановку приоритетов элементов беклога.
- Управление беклогом продукта включает в себя:
  - описание элементов беклога ясным и понятным образом;
  - управление порядком элементов беклога для наилучшего достижения целей и миссий;
  - оптимизацию ценности работы, выполняемой разработчиками;
  - обеспечение доступности, прозрачности и ясности беклога для всех участников процесса.
  - понимание разработчиками элементов беклога.
- Его решения по исполнению тех или иных задач должны выполняться.
- Никто не может заставить команду разработки работать над другим набором требований.
- Он не может влиять на работу команды разработки, однако он всегда должен находиться рядом для разрешения возникающих вопросов.

Роли в Scrum: Scrum Мастер

- обучение команды особенностям Scrum–методологии;
- устранение проблем, образующихся внутри команды;
- выявление рисков и проблем, устранение препятствий, мешающих прогрессу работы;
- создание дружественных отношений в команде;
- слежение за процессами и их выполнением (разработки, тестирования);
- смена статусов задач в спринте;
- проведение ежедневных летучек;
- организация встреч перед спринтами;
- помощь владельцу продукта с беклогом продукта.

Услуги Scrum Мастера:

Владелец продукта	Команда разработке	Организация в целом
Помогает найти наиболее эффективные методы для управления Бэклогом Продукта	Обучает команду самоорганизации и многофункциональности	Адаптирует Scrum в соответствии с потребностями организации
По необходимости помогает в организации процессов	Устраняет внешние препятствия	Помогает понять внешним к команде людям Scrum
При необходимости может выступить ведущим мероприятий Скрама.	По необходимости помогает а организации процессов	Обменивается лучшими практиками с другими Scrum Мастерами

Роли в Scrum: Команда разработки

- Самоорганизующаяся и самоуправляемая команда.
- Размер команды в идеале составляет от 5 до 9 человек.
- Команда разработчиков – основная движущая сила, которая выполняет все технические задачи по разработке.
- Никто (и даже Скрам Мастер) не может указывать команде, как правильно превращать беклог продукта в инкременты работающей функциональности.
- Каждый член команды имеет специализированные знания, но ответственность всегда на всей команде в целом.
- В команде есть только понятие разработчик продукта и исключены все должности. Всё это, не смотря на то, чем занимается конкретный человек.
- Команда кроссфункциональна, и обладают всеми навыками, необходимыми для разработки инкремента продукта.
- Команда не имеет иерархии или подотделов. Всё решается внутри команды.

Роли в Scrum: Команда разработки

В ключевые обязанности команды входят:

- занимается непосредственной разработкой для заказчика;
- следит за собственной результативностью (совместно с Scrum Master);
- берет на себя решение по разработке и дизайну;
- создает беклог спринта;
- обеспечивает в конце спринта реализацию потенциально поставляемой функциональности;
- оценивает элементы беклога продукта;
- несет ответственность за результат перед владельцем продукта.

Команда должна быть уполномочена определять:

- что она будет совершать в конце спринта;
- как ожидаемые результаты должны быть разбиты на задачи;
- кто выполнит задачу и в каком порядке они будут выполнены.

Слабые стороны Scrum

- **Высокая требовательность к команде**: кроссфункциональность, члены команды должны быть «командными игроками», активно брать на себя ответственность и уметь самоорганизовываться. Подобрать такую зрелую команду очень непросто!.
- **Чрезмерная ориентированность на набор очков**. Разработчикистараются набрать побольше очков во время спринта, что не приводит к улучшению кодовой базы, не делает её проще.
- **Длительные митапы**. Работникам приходится встраивать многочасовые совещания в свое расписание.
- **Неизменность элементов беклога во время спринта**. У программистов нет возможности перераспределить работу при обнаружении новых особенностей. Scrum не позволяет перестраивать корабль прямо во время плавания, поэтому приходится ждать окончания спринта, чтобы внести изменения.

Kanban

- Kanban очень похож на схему промышленного производства. На входе в этот процесс попадает кусочек металла, а на выходе получается готовая деталь.
- Также и в Kanban, инкремент продукта передаётся вперёд с этапа на этап, а в конце получается готовый к поставке элемент.
- Создатель Kanban вдохновлялся супермаркетами, а именно их принципом – «держи на полках только то, что нужно клиенту». А потому в Kanban разрешается оставить неоконченную задачу на одном из этапов, если её приоритет изменился и есть другие срочные задачи. Неотредактированная статья для блога, подвешенная без даты публикации или часть кода функции, которую возможно не будут включать в продукт – всё это нормально для работы по Kanban.
- Kanban намного менее строгий, нежели Scrum – он не ограничивает время спринтов, нет ролей, за исключением владельца продукта. Kanban даже позволяет члену команды вести несколько задач одновременно, чего не позволяет Scrum. Также никак не регламентированы встречи по статусу проекта – можно делать это как Вам удобно, а можно не делать вообще.

Схема работы по Kanban

- Для работы с Kanban необходимо определить этапы потока операций. В Kanban они изображаются как столбцы, а задачи обозначают специальные карточки.
- Карточка перемещается по этапам, подобно детали на заводе, переходящей от станка к станку, и на каждом этапе процент завершения становится выше. На выходе мы получаем готовый к поставке заказчику элемент продукта.
- Доска со столбцами и карточками может быть как настоящей, так и электронной.

4 столпа Kanban

- **Карточки.** Для каждой задачи создаётся индивидуальная карточка, в которую заносится вся необходима информация о задаче. Таким образом, вся нужная информация о задаче всегда под рукой.
- **Ограничение на количество задач на этапе.** Количество карточек на одном этапе строго регламентировано. Благодаря этому сразу становится видно, когда в потоке операций возникает «затор», который оперативно устраняется.
- **Непрерывный поток.** Задачи из беклога попадают в поток в порядке приоритета. Таким образом, работа никогда не прекращается.
- **Постоянное улучшение**: Концепция постоянного улучшения появилась в Японии в конце XX века. Её суть в постоянном анализе производственного процесса и поиске путей повышения производительности.

Отличие Scrum и Kanban

	Скрам	Канбан
Темп	Повторяемые спринты фиксированной продолжительности	Непрерывный процесс
Выпуск релиза	В конце каждого спринта после одобрения проектным менеджером (владельцем продукта)	Поток продолжается без перерывов или на усмотрение команды
Роли	Владелец продукта, Scrum-мастер, команда разработчиков	Команда под руководством ПМ, в некоторых случаях привлекаются тренеры по agile kanban
Главные показатели	Скорость команды	Ведущее время
Приемлемость изменений	В ходе спринта изменения нежелательны, так как могут привести к неверной оценке задач	Изменение могут случиться в любой момент

Недостатки Kanban

- Система плохо работает с командами численностью более 5 человек.
- Не предназначен для долгосрочного планирования.
- Система выливается в нескончаемый поток задач.

Extreme Programming

- Особенность Scrum заключается в том, что этот фреймворк уделяет мало внимания практикам разработки. Поэтому некоторые agile-компании (порядка 10%) комбинируют его с экстремальным программированием (XP).
- Экстремальное программирование привлекло к себе внимание в конце 90-х. Концепция зародилась в сообществе Smalltalk.
- Первым проектом, созданным по методологии Extreme Programming, стала система контроля платежей Chrysler Comprehensive Compensation (C3) в середине девяностых.
- Сам термин «экстремальное программирование» появился в 1997 году.

Практики Extreme Programming

Управленческие практики

- Игра в планирование
- Частые небольшие релизы
- Заказчик всегда рядом
- 40-часовая рабочая неделя
- Коллективное владение кодом

Инженерные практики

- Непрерывная интеграция
- Парное программирование
- Разработка через тестирование
- Рефакторинг
- Простота
- Метафора системы
- Стандарт кодирования

Extreme Programming

- XP сильно напоминает Scrum и предполагает, что заказчик плотно взаимодействует с командой разработчиков, расставляя приоритеты (истории). Программисты также оценивают, планируют и реализуют задачи короткими итерациями.
- Однако Extreme Programming делает сильный упор на тестирование, что отличает её от Scrum. Методология гласит, что разработчик не может сказать, что код написан правильно до тех пор, пока не пройдут все модульные тесты.
- Но такая «тестоориентированность» одновременно и недостаток подхода. Чтобы адаптировать XP, нужно инвестировать в создание инфраструктуры автоматизированных тестов и непрерывного развертывания ПО.
- Если в случае Scrum компания может послать менеджеров проектов на двухдневные курсы, то в случае с экстремальным программированием приходится тренировать всю команду разработчиков. Что гораздо более затратно.

Статистика применения гибких технологий по результатам исследования Agile Survey