Manual QA course

Lecture 18. Методологии разработки ПО.

Дорофеев Максим

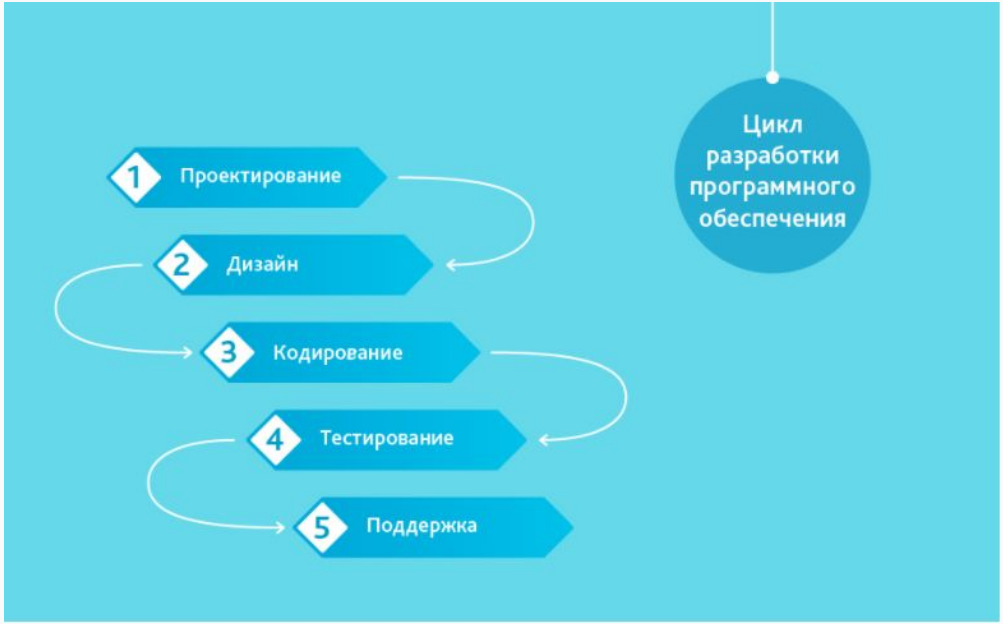
hillel
компьютерная
школа

Waterfall.

Самая старая методология разработки ПО. Описана в 1970 году Винстоном Ройсом.

Подразумевает последовательное прохождение всех стадий, каждая из которых должна полностью завершиться до начала предыдущей.

Waterfall.


Цикл разработки программного обеспечения
1 Проектирование
2 Дизайн
3 Кодирование
4 Тестирование
5 Поддержка

Waterfall. Плюсы.

- Легок для понимания и использования;
- Детально структурирован;
- Задает стабильные требования к продукту;
- Проекты легко контролируются;
- Качество имеет первоочередный приоритет.

Waterfall. Минусы.

- Все требования должны быть определены до начала разработки;
- Дорого и медленно;
- Чувствителен к изменениям;
- Мало возможностей повлиять на цели проекта;
- Зачастую проблемы выявляются на этапе тестирования;
- Много документации (в том числе и технической), которая непонятна конечному пользователю и заказчику.

Waterfall. Когда применять?

- Требования к продукту предельно ясны и стабильны;
- Известны используемые технологии и инструменты;
- Проект большой, дорогой, сложный;

Примеры:

- ПО для адронного коллайдера;
- ПО для космической промышленности.

Agile.

Гибкая методология разработки (англ. Agile software development, agile-методы) — серия подходов к разработке программного обеспечения, которая характеризуется акцентом на людях и их взаимодействии, рабочем продукте, а также на гибкости и реагировании на изменения.

Agile.

В основе гибкой методологии лежит итеративный процесс разработки. Работа над проектом проходит циклами (длительность 1-4 недели) и подразумевается что в конце каждого цикла выпускается новая готовая версия продукта. Дальше идет анализ полученных результатов и планируется работа над следующим циклом, и так далее

Agile vs Waterfall.

Agile методология является альтернативой **waterfall** модели (**водопадный** или **каскадный** процесс разработки).

Agile vs Waterfall.

В каскадной модели есть ряд определенных этапов процесса разработки и разработчик последовательно переходит от одного этапа к другому. Каскадная модель разработки подразумевает, что переход к новому этапу разработки начинается только после полного завершения предыдущей

Agile vs Waterfall.

Agile. Начало.

Активное вхождение Agile в массы началось после подписания Agile Manifesto 11-13 февраля 2001 года на лыжном курорте в штате Юта США. Этот манифест подписали представители таких методологий как Scrum, Crystal Clear, Extreme Programming, Adaptive Software Development, Feature Driven Development, and Dynamic Systems Development Method (DSDM)

Agile. Почему появился?

- Заказчик не может сформировать четкие требования;
- Новые технологии усилили конкуренцию в бизнесе;
- Заказчики и разработчики не удовлетворены процессом взаимодействия;
-

Agile. Основные идеи.

- Люди и взаимодействие важнее процессов и инструментов;
- Работающий продукт важнее исчерпывающей документации;
- Сотрудничество с заказчиком важнее согласования условий контракта;
- Готовность к изменениям важнее следования первоначальному плану.

Agile vs Waterfall.

То есть, не отрицая важности того, что справа, мы всё таки больше ценим то, что слева

Agile Manifesto.

Особенность в том, что в манифесте не описаны какие либо действия или правила, а описаны только основные принципы, на базе которых может строиться подход к разработке.

Agile Manifesto. Принципы.

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

Agile Manifesto. Принципы.

Изменение требований приветствуется, даже на поздних стадиях разработки.

Agile Manifesto. Принципы.

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

Agile Manifesto. Принципы.

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

Agile Manifesto. Принципы.

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

Agile Manifesto. Принципы.

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

Agile Manifesto. Принципы.

Работающий продукт — основной показатель прогресса.

Agile Manifesto. Принципы.

Инвесторы, разработчики и пользователи должны иметь возможность поддерживать постоянный ритм бесконечно.

Agile Manifesto. Принципы.

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

Agile Manifesto. Принципы.

Простота - искусство не делать лишней работы

Agile Manifesto. Принципы.

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

Agile Manifesto. Принципы.

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

Agile Manifesto. Принципы.

1. Наивысшим приоритетом является удовлетворение потребностей заказчика, благодаря регулярной и ранней поставке ценного программного обеспечения.
2. Изменение требований приветствуется, даже на поздних стадиях разработки.
3. Работающий продукт следует выпускать как можно чаще, с периодичностью от пары недель до пары месяцев.
4. На протяжении всего проекта разработчики и представители бизнеса должны ежедневно работать вместе.
5. Над проектом должны работать мотивированные профессионалы. Чтобы работа была сделана, создайте условия, обеспечьте поддержку и полностью доверьтесь им.
6. Непосредственное общение является наиболее практичным и эффективным способом обмена информацией как с самой командой, так и внутри команды.
7. Работающий продукт — основной показатель прогресса.
8. Инвесторы, разработчики и пользователи должны иметь возможность поддерживать постоянный ритм бесконечно.
9. Постоянное внимание к техническому совершенству и качеству проектирования повышает гибкость проекта.
10. Простота — искусство минимизации лишней работы — крайне необходима.
11. Самые лучшие требования, архитектурные и технические решения рождаются у самоорганизующихся команд.
12. Команда должна систематически анализировать возможные способы улучшения эффективности и соответственно корректировать стиль своей работы.

Agile. Преимущества.

Частые релизы — требования не успевают устаревать, частью функционала уже можно пользоваться;
Фиксированная длина итераций — можно предсказывать скорость работы команды с учетом рисков;
Команда сама оценивает задачи — оценки реалистичны, команда мотивирована выполнить свои обязательства;
Команда самоуправляемая — 10 голов учтут больше чем одна очень умная;
В конце каждой итерации процесс работы оценивается и вносятся улучшения;
Команда кроссфункциональная — границы отделов компании не являются препятствием при сотрудничестве, разнообразные навыки сочетаются и происходит синергия.

Agile. Применение.

Спектр применения **Agile** довольно широк: от небольших студенческих стартапов, до крупных промышленных проектов размером в тысячи человеко-часов, как в локальной команде, так и в проекте с географически распределенными командами.

Agile. Применение.

Не каждой команде может подойти применение гибкой методологии. Для некоторых проектов будет удачной и водопадная модель.

Agile. Применение

Не всегда применение гибких методологий может дать положительный эффект. Agile может негативно сказаться на эффективности:

- В проектах, которые являются инфраструктурными и имеют очень сложный процесс поддержки;
- В проектах, где подтверждение технического задания требует очень длительного формального цикла;
- Если нет обратной связи с конечным пользователем системы или нет возможности у команды провести экспертизу в предметной области;
- Если команда состоит из недостаточно квалифицированных специалистов, которые не готовы к изменениям и внедрению прогрессивных подходов.

Agile. Scrum.

Scrum (от англ. scrum «толкучка») — методология управления проектами, активно применяющаяся при разработке информационных систем для гибкой разработки программного обеспечения. Scrum чётко делает акцент на качественном контроле процесса разработки. Кроме управления проектами по разработке ПО, Scrum может также использоваться в работе команд поддержки программного обеспечения (software support teams), или как подход управления разработкой и сопровождением программ: Scrum of Scrums.

Agile. Scrum.

Скрам (Scrum) — это набор принципов, на которых строится процесс разработки, позволяющий в жёстко фиксированные и небольшие по времени итерации, называемые спринтами (sprints), предоставлять конечному пользователю работающее ПО с новыми возможностями, для которых определён наибольший приоритет. Возможности ПО к реализации в очередном спринте определяются в начале спринта на этапе планирования и не могут изменяться на всём его протяжении. При этом строго фиксированная небольшая длительность спринта придаёт процессу разработки предсказуемость и гибкость.

Agile. Scrum.

Спринт — итерация в скраме, в ходе которой создаётся функциональный рост программного обеспечения. Жёстко фиксирован по времени. Длительность одного спринта от 2 до 4 недель.

Agile. Scrum.

В отдельных случаях, к примеру согласно **скрам-стандарту** компании **Nokia**, длительность спринта должна быть не более 6 недель. Тем не менее, считается, что чем короче спринт, тем более гибким является процесс разработки, релизы выходят чаще, быстрее поступают отзывы от потребителя, меньше времени тратится на работу в неправильном направлении.

Agile. Scrum.

С другой стороны, при более длительных спринтах команда имеет больше времени на решение возникших в процессе проблем, а владелец проекта уменьшает издержки на совещания, демонстрации продукта и т. п. Разные команды подбирают длину спринта согласно специфике своей работы, составу команд и требований, часто методом проб и ошибок. Для оценки объёма работ в спринте можно использовать предварительную оценку, измеряемую в очках истории. Предварительная оценка фиксируется в бэклоге проекта.

Agile. Scrum.

Беклог проекта (англ. Project backlog) — это список требований к функциональности, упорядоченный по их степени важности, подлежащих реализации. Элементы этого списка называются пользовательскими историями (user story) или элементами беклога (backlog items). Беклог проекта открыт для редактирования для всех участников скрам-процесса.

Agile. Scrum.

Беклог спринта (англ. Sprint backlog) — содержит функциональность, выбранную владельцем проекта из беклога проекта. Все функции разбиты по задачам, каждая из которых оценивается скрам-командой. Каждый день команда оценивает объём работы, который нужно проделать для завершения спринта.

Agile. Scrum.

Диаграмма сгорания задач (Burndown chart) - диаграмма, показывающая количество сделанной и оставшейся работы. Обновляется ежедневно с тем, чтобы в простой форме показать подвижки в работе над спринтом. График должен быть общедоступен.

Agile. Scrum.

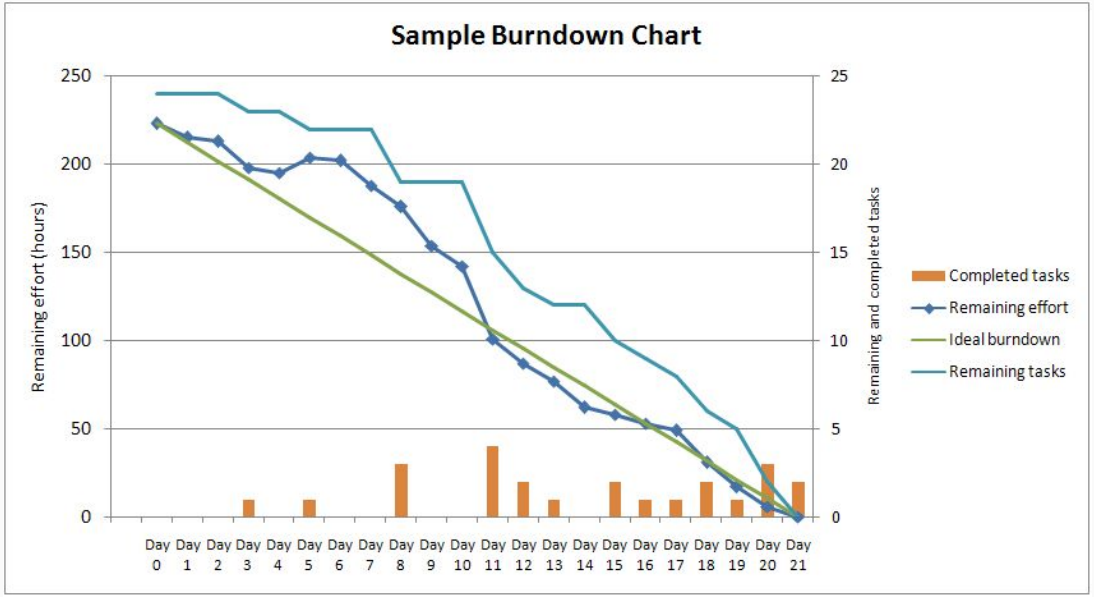
Существуют разные виды диаграммы:

Диаграмма сгорания работ для спринта — показывает, сколько уже задач сделано и сколько ещё остаётся сделать в текущем спринте.

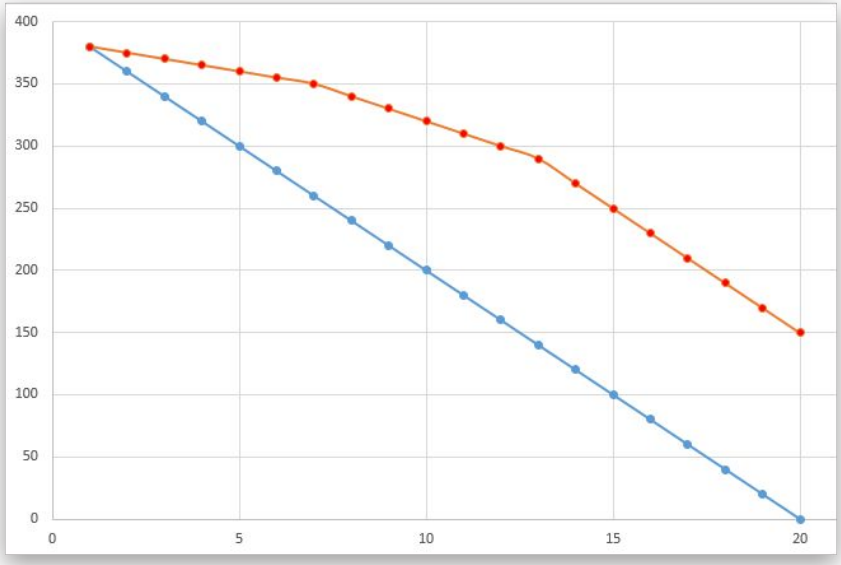
Диаграмма сгорания работ для выпуска проекта — показывает, сколько уже задач сделано и сколько ещё остаётся сделать до выпуска продукта (обычно строится на базе нескольких спринтов).

Agile. Scrum.

Диаграмма отображает завершенный спринт. Показывает оставшиеся нерешённые задачи и трудозатраты, необходимые для их завершения в расчёте на 21 рабочий день.



Agile. Scrum. Burndown chart.



Agile. Scrum. Burndown chart.

Agile. Scrum.

История спринта (Sprint Story) - требуемую функциональность, которую добавляют в бэклог, часто называют историей. Зачастую история имеет следующую структуру: «Будучи пользователем <тип пользователя> я хочу сделать <действие>, чтобы получить <результат>». Такая структура удобна тем, что понятна как разработчикам так и заказчикам.

Остановка спринта (Abnormal Termination) - остановка спринта может быть произведена раньше срока его планового окончания в исключительных ситуациях. Спринт может остановить команда, если понимает, что не может достичь цели спринта в отведенное время. Спринт может остановить владелец проекта, если исчезает необходимость в реализации цели спринта. После остановки спринта проводится совещание с командой, где обсуждаются причины остановки. После этого начинается новый спринт.

Покер планирование (Planning Poker)

Очки за пользовательскую историю (Story Points) - абстрактная метрика оценки сложности истории, которая не учитывает затраты в человеко-часах. Обычно используют одну из следующих шкал: ряд Фибоначчи (1,2,3,5,8,13,21,34,55); линейную шкалу (1,2,3,4 … n); степень двойки (1,2,4,8 … 2n); размеры одежды (XS, S, M, L, XL).

Задачи истории спринта (Sprint Story Tasks) - добавляются к историям спринта. Выполнение каждой задачи оценивается в часах. Каждая задача не должна превышать 12 часов (зачастую команда настаивает, чтобы максимальная продолжительность задачи равнялась одному рабочему дню).

Критерий готовности (Definition of Done (DoD)) - критерии, определяющие степень готовности элемента из журнала пожеланий пользователя.

Скорость команды (Velocity) - общее количество очков, набранных командой за предыдущий спринт. Данная метрика помогает команде понять, сколько историй она может сделать за один спринт.

Agile. Scrum. Ритуалы. Sprint planning meeting.

- Выполняется всей командой перед началом спринта;
- Команда выбирает требования из Product backlog и формирует Sprint backlog;
- Команда декомпозирует требования на задачи;
- Каждая задача проходит оценку;
- Во время встречи PO отвечает на вопросы команды.

Agile. Scrum. Ритуалы. Sprint planning meeting. Структура.

- Представление и пояснение PO списка требований;
- Вопросы со стороны команды;
- (Перерыв);
- Декомпозиция требований на задачи;
- Оценка задач.

В начале проекта может занимать 5 -6 часов. Но со временем будет более оперативной, примерно 2 - 3 часа.

Agile. Scrum. Ритуалы. Daily meeting.

- Проходит ежедневно в одно и тоже время, и в одном и том же месте;
- встреча проходит только стоя;
- Длительность не более 15 минут;
- Каждый должен всего на **3 вопроса**: Что делал вчера? Что буду делать сегодня? Какие есть проблемы?

Agile. Scrum. Ритуалы. Sprint review.

По завершению каждого спринта команда должна провести демонстрацию полученного результата.

Agile. Scrum. Ритуалы. Sprint review.

- Команда зачитывает требования из **Sprint backlog**;
- По каждому критерию приемки происходит демонстрация полученного результата;
- Каждый вопрос **PO**, записывается для ответа на них позже;
- Каждое новое требование **PO**, записывается, для включения его в **Product backlog**.

Agile. Scrum. Ритуалы. Retrospective.

Ритуал направлен на обмен опытом внутри команды, проводится после **Sprint review.**

Agile. Scrum. Ритуалы. Retrospective.

- Какие решения должна принять команда, чтобы сделать процесс более предсказуемым;
- какие проблемы мешают выполнять взятые на себя обязательства;
- Как улучшить взаимодействие с PO;
- Какие ошибки совершает команда и почему.

Agile. Scrum. Roles.

Свинья идёт по дороге. Курица смотрит на неё и говорит: «А давай откроем ресторан!» Свинья смотрит на курицу и отвечает: «Хорошая идея, и как ты хочешь его назвать?» Курица думает и говорит: «Почему бы не назвать „Яичница с беконом“?». «Так не пойдёт, — отвечает свинья, — ведь тогда я полностью посвящу себя проекту, а ты будешь вовлечена только частично».

Agile. Scrum. Roles.

«Свиньи» полностью включены в проект и в скрам-процесс:

Скрам-мастер (Scrum Master) — проводит совещания (Scrum meetings) следит за соблюдением всех принципов скрам, разрешает противоречия и защищает команду от отвлекающих факторов. Данная роль не предполагает ничего иного, кроме корректного ведения скрам-процесса. Руководитель проекта скорее относится к владельцу проекта и не должен фигурировать в качестве скрам-мастера.

Владелец продукта (Product Owner) — представляет интересы конечных пользователей и других заинтересованных в продукте сторон.

Скрам-команда (Scrum Team) — кросс-функциональная команда разработчиков проекта, состоящая из специалистов разных профилей: тестировщиков, архитекторов, аналитиков, программистов и т. д. Размер команды в идеале составляет от 3 до 9 человек. Команда является единственным полностью вовлеченным участником разработки и отвечает за результат как единое целое. Никто, кроме команды не может вмешиваться в процесс разработки на протяжении спринта.

Agile. Scrum. Roles. Product owner.

- Формирует требования;
- Приоритезирует требования;
- Корректирует приоритеты на каждом спринте;
- Несет персональную ответственность за ценность требований;
- Отвечает за взаимодействие с рынком;
- Только один человек.

Agile. Scrum. Roles. Scrum Master.

- Следит за корректным применением Agile - принципов;
- Организует работу команду и обеспечивает ее всем необходимым;
- Защищает команду, несет ответственность за ее эффективность;
- Только один человек.

Agile. Scrum. Roles. Scrum Master.

- Не назначает людей на задачи - это делает сама команда;
- Не заставляет делать людей работу - это ответственность команды;
- Не указывает PO, какие требования писать к продукту - это ответственность PO.

Agile. Scrum. Roles. Team.

- Кросс - функциональная;
- Взаимозаменяемая;
- Самоорганизующаяся;
- С фиксированным составом (в ходе спринта);
- 4 - 8 человек.

Agile. Scrum. Roles. Team.

- Определяет продолжительность спринта;
- Емкость команды;
- Трудоемкость требований, которые будут реализованы в спринте;
- Очередность выполнения задач.

Agile. Scrum. Roles. Дополнительные роли (Ancillary roles) в методологии скрам.

«Куры»

- Пользователи (Users)
- Клиенты, Продавцы (Stakeholders) — лица, которые инициируют проект и для кого проект будет приносить выгоду. Они вовлечены в скрам только во время обзорного совещания по спринту (Sprint Review).
- Управляющие (Managers) — люди, которые управляют персоналом.
- Эксперты-консультанты (Consulting Experts)

Agile. Scrum.

Планирование спринта (Sprint Planning Meeting)
Происходит в начале новой итерации Спринта.

- Из бэклога проекта выбираются задачи, обязательства по выполнению которых за спринт принимает на себя команда;
- На основе выбранных задач создается бэклог спринта. Каждая задача оценивается в идеальных человеко-часах;
- Решение задачи не должно занимать более 12 часов или одного дня. При необходимости задача разбивается на подзадачи;
- Обсуждается и определяется, каким образом будет реализован этот объём работ;
- Продолжительность совещания ограничена сверху 4-8 часами в зависимости от продолжительности итерации, опыта команды и т. п.
- (первая часть совещания) Участвует владелец проекта и скрам команда: выбирают задачи из бэклога продукта;
- (вторая часть совещания) Участвует только команда: обсуждают технические детали реализации, наполняют бэклог спринта

Agile. Scrum.

Ежедневное совещание (Daily Scrum meeting)
- начинается точно вовремя;
- все могут наблюдать, но только «свиньи» говорят;
- длится не более 15 минут;
- проводится в одном и том же месте в течение спринта.

В течение совещания каждый член команды отвечает на 3 вопроса:
- Что я сделал с момента прошлой встречи для того, чтобы помочь Команде Разработки достигнуть Цели Спринта?
- Что я сделаю сегодня для того, чтобы помочь Команде Разработки достичь Цели Спринта?
- Вижу ли я препятствия для себя или Команды Разработки, которые могли бы затруднить достижение Цели Спринта? (Над решением этих проблем работает скрам мастер. Обычно это решение проходит за рамками ежедневного совещания и в составе лиц, непосредственно затронутых данным препятствием.)

Agile. Scrum.

Скрам над скрамом (Scrum of Scrums)
Проводится после ежедневного скрам совещания. Позволяет нескольким скрам командам обсуждать работу, фокусируясь на общих областях и взаимной интеграции. Повестка та же, что и на ежедневном скрам совещании плюс следующие вопросы:

- Что каждая команда сделала с момента предыдущего ежедневного совещания?
- Что каждая команда сделает к следующему ежедневному совещанию
- Есть ли проблемы, мешающие или замедляющие работу каждой команды?
- Нужно ли другой команде сделать что-то из задач вашей команды?

Agile. Scrum.

Обзор итогов спринта (Sprint review meeting)
Проводится в конце спринта.

- Команда демонстрирует прирост функциональности продукта 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.
- Ограничена четырьмя часами в зависимости от продолжительности итерации и прироста функциональности продукта.

Agile. Scrum.

Ретроспективное совещание (Retrospective meeting)
Проводится в конце 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).
Ограничена одним-тремя часами.

Kanban.

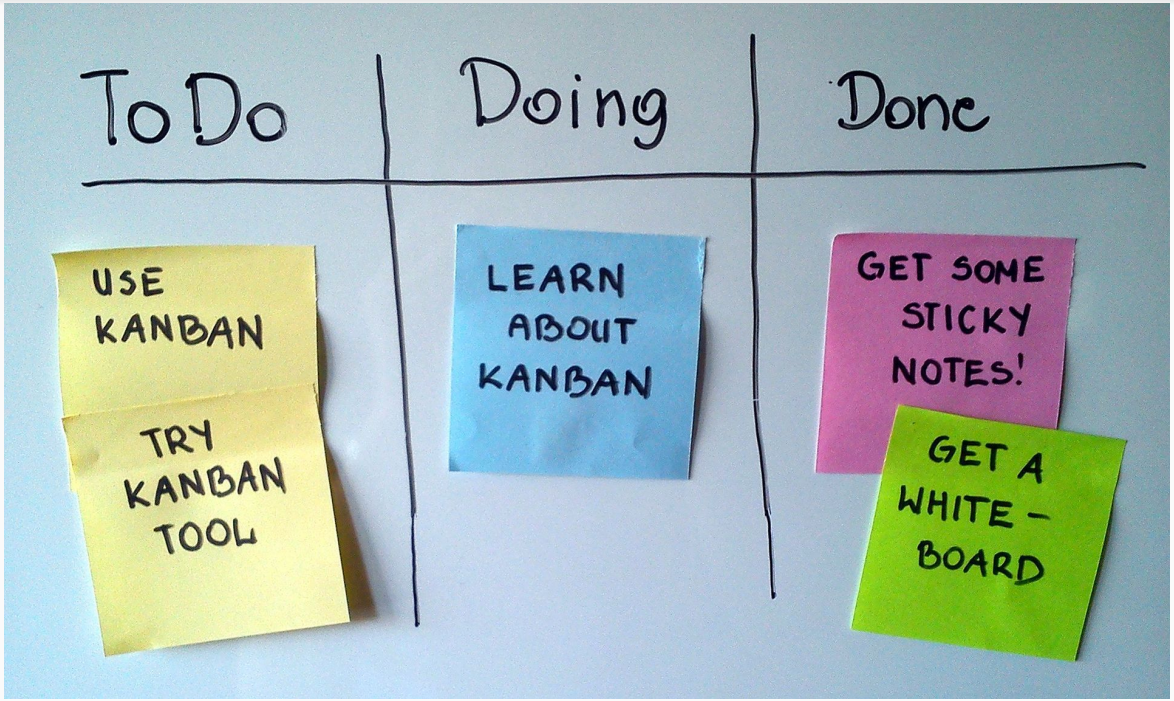
Канбан — метод управления разработкой, реализующий принцип «точно в срок» и способствующий равномерному распределению нагрузки между работниками. При данном подходе, весь процесс разработки прозрачен для всех членов команды. Задачи по мере поступления заносятся в отдельный список, откуда каждый разработчик может извлечь требуемую задачу.

Kanban.

Канбан основан на четырех основных принципах:

Опора на существующие методы разработки
Канбан начинается с существующих методов разработки и стимулирует в них дополнительные изменения.
Предварительная договоренность о проведении важных изменений
Команда разработчиков должна учитывать, что постоянные изменения — это способ улучшить существующий процесс разработки, однако проведение глобальных перемен имеет большой риск. Канбан поощряет небольшие и эволюционные изменения.
Уважение к существующему порядку, ролям и обязанностям
Поощрение инициативы
Приветствуются проявления инициативы каждого разработчика.

Kanban.


To Do
Doing
Done
USE KANBAN
TRY KANBAN TOOL
LEARN ABOUT KANBAN
GET SOME STICKY NOTES!
GET A WHITE-BOARD

Kanban.

Основные принципы Канбан Доски:

- Визуализация рабочего процесса;
- Ограничение работы, которая находится в процессе;
- Перемещение задач от колонки к колонке;
- Мониторинг, адаптация и оптимизация;

Kanban.

- Отменяется разработка по фазам;
- Пользовательские истории больше, но их меньше;
- Оценка задач сводится к минимуму или убирается совсем;
- Внимание переходит со скорости разработки на продолжительность цикла.

Вопросы и ответы