Методологии разработки ПО

Что будем изучать?

- Ключевые роли в проектной команде
- Основные модели жизненного цикла разработки ПО
- Методология Agile

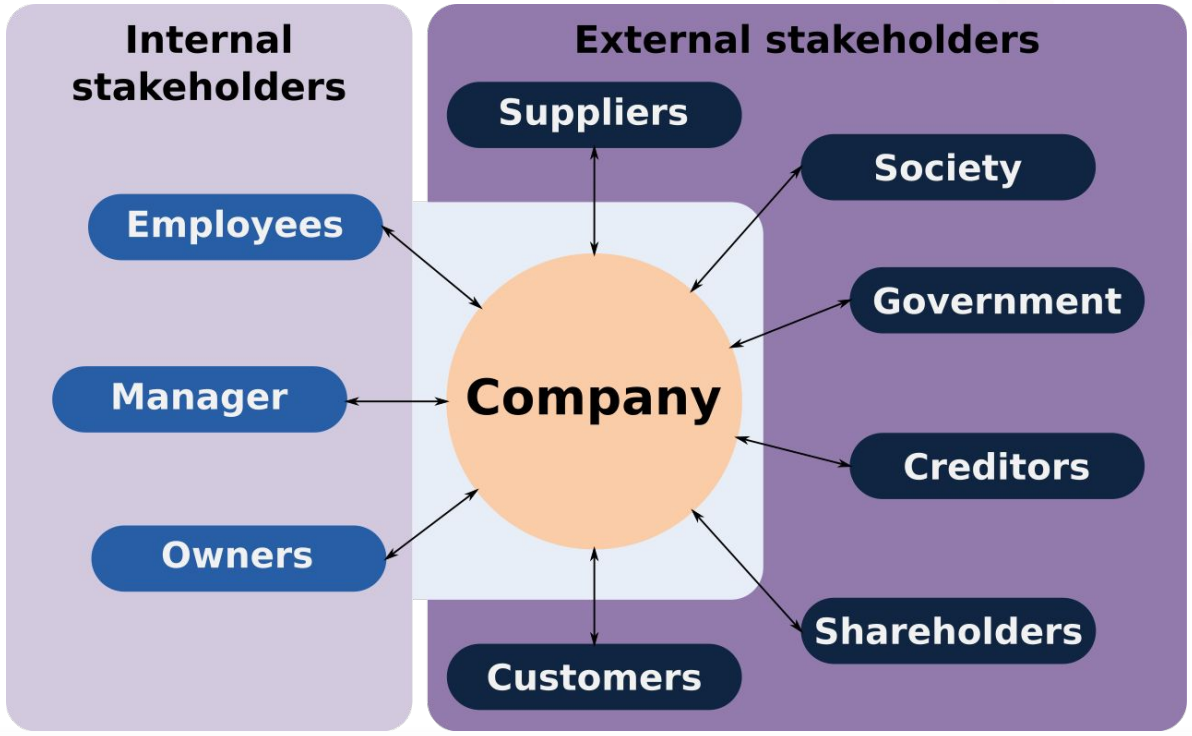
Ключевые роли в проектной команде

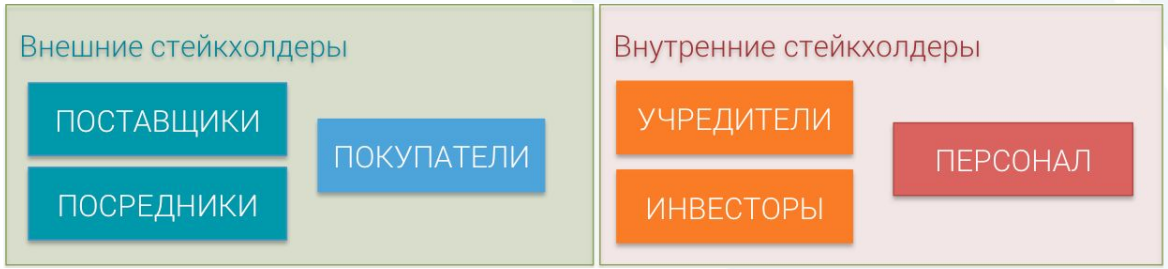
- Stakeholders (стейкхолдер)
- Project Manager (проектный менеджер)
- Business analyst (бизнес аналитик)
- UI/UX designer (UI/UX дизайнер)
- Developers (разработчики)
- QA

A-Level
UKRAINE

Stakeholders

Stakeholders (заинтересованная сторона, причастная сторона) – понятие, которое описывает человека, группу лиц или отдельные организации, чьи действия, поведение или решения могут влиять на прибыль компании и процессы в ней. Стейкхолдеров разделяют на внутренних (находятся внутри организации) и внешних (за пределами предприятия).





Теория управления стейкхолдерами

A-Level UKRAINE

Project Manager (PM)

Project Manager – это специалист, чьей главной задачей является управление проектом в целом: проектирование и расстановка приоритетов, планирование выполнения задач, контроль, коммуникации, а также оперативное решение проблем.

Основная обязанность и ответственность PM – довести идею заказчика до реализации в установленный срок, используя существующие ресурсы. В рамках этой задачи PM'у необходимо построить план разработки, организовать команду, настроить процесс работы над проектом, обеспечить обратную связь между командами и заказчиком, устранять помехи для команд, контролировать качество и поставку продукта в срок.

Карьера в IT: должность Project Manager

Обязанности PM’а:

— составление плана проекта и ведение проектной документации;
— согласование сроков и анализ возможный рисков;
— участие в подборе и утверждении проектной команды;
— определение требуемых ресурсов и рабочей среды, их распределение внутри команды;
— постановка рабочего процесса в команде (разработка, тестирование, работа с требованиями);
— определение и отслеживание должной приоритетности выполнения задач;
— отслеживание нагрузки задачами и прогресса по задачам каждого разработчика;
— отслеживание сроков выполнения задач;
— удерживание команды в рабочем состоянии, мотивация команды;
— создание прозрачной среды общения между всеми участниками процесса;
— отслеживание удовлетворенности проектом со стороны команды;
— решение всевозможных конфликтных ситуаций внутри команды и в связке заказчик-команда;
— общение с заказчиком, управление его ожиданиями;
— предоставление заказчику отчетности о ходе выполнения задач и проекта в целом;
— презентация заказчику готовых решений, демо-версий, прототипов.

Business Analyst (BA)

Бизнес-аналитик – это специалист, который исследует проблему заказчика, ищет решение и оформляет его концепцию в форме требований, на которые в дальнейшем будут ориентироваться разработчики при создании продукта.

Карьера в IT: должность Бизнес-аналитик

В круг обязанностей бизнес-аналитика входит:

- Анализ бизнес-потребностей заказчика;
- Составление требований к будущему продукту (общение с заинтересованными лицами – разработчиками, клиентами, конечными пользователями);
- Анализ требований (применение различных методологий и нотаций – прототипирование, анкетирование, опрос, мозговой штурм, анализ существующих документаций, конкурентов);
- Анализ проблемных областей и предложения для улучшения;
- Формализация требований (разделение требований на функциональные, не функциональные, написание спецификации требований);
- Управление требованиями (обработка запросов на изменение, анализ и описание влияния на существующие требования);
- Трансляция требований между разработчиками и клиентом.

UI/UX дизайнер

UI/UX дизайнер – это креативный специалист, который проектирует пользовательские интерфейсы. UI и UX – это два разных профиля дизайна, но чаще всего задачи по обоим направлениям тесно связаны между собой, а потому их делает один универсальный специалист.

UI (User Interface) – пользовательский интерфейс

UX (User experience) – пользовательский опыт

Карьера в IT: должность UX/UI дизайнер

UI-дизайнер отвечает за визуализацию приложения, делая его удобным и функциональным. Дабы продукт с комфортом воспринимался глазами пользователя, специалист UI отвечает за подбор форм, цветов и других параметров. Что касается UX-дизайнера, он в большей степени ответственен за функциональность дизайна. В итоге: приложением легко и удобно пользоваться.

Ключевые обязанности UX/UI дизайнера:

- Сбор информации о проекте и его аудитории;
- Проектирование пользовательских сценариев;
- Разработка стиля, составление инструкций по шрифтам, цветам и размерам;
- Создание макетов и прототипов;
- Отрисовка интерфейса в графических редакторах.

Developers

К масштабным проектам, где требуется техническое управление в первую очередь подключается руководитель группы разработчиков (Team Lead).

В числе его задач:

- Формирование сплоченной команды, ее микроклимата и корпоративной культуры;
- Определение стратегии разработки: формирование кодстайла, достижение запланированной производительности, обеспечение требований безопасности, выбор правильного архитектурного решения;
- Распределение задач между членами команды, контроль их выполнения, соблюдение сроков и других требований, проведение кодревью.

A-Level UKRAINE

Qubit LABS

Front-end

Front-end devs take care of the user interface, its usability, architecture, and visual appeal.

Back-end

The invisible part of the process that gives you a fast and relevant answer is called back-end.

Full-stack

Full-stack development is the coding of the website, app, or software product by one person combining the functions of back-end and front-end developer.

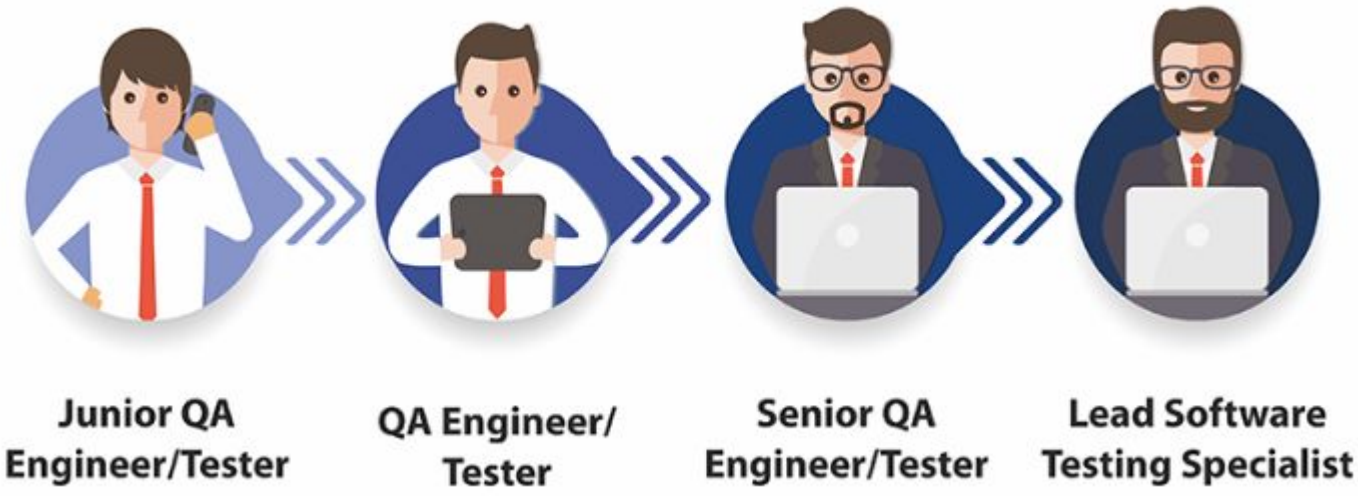
A-Level UKRAINE

QA engineer

QA engineer – это специалист по обеспечению качества, деятельность которого направлена на улучшение процесса разработки ПО, предотвращение дефектов и выявление ошибок в работе продукта.

Процесс обеспечения качества состоит из таких этапов:

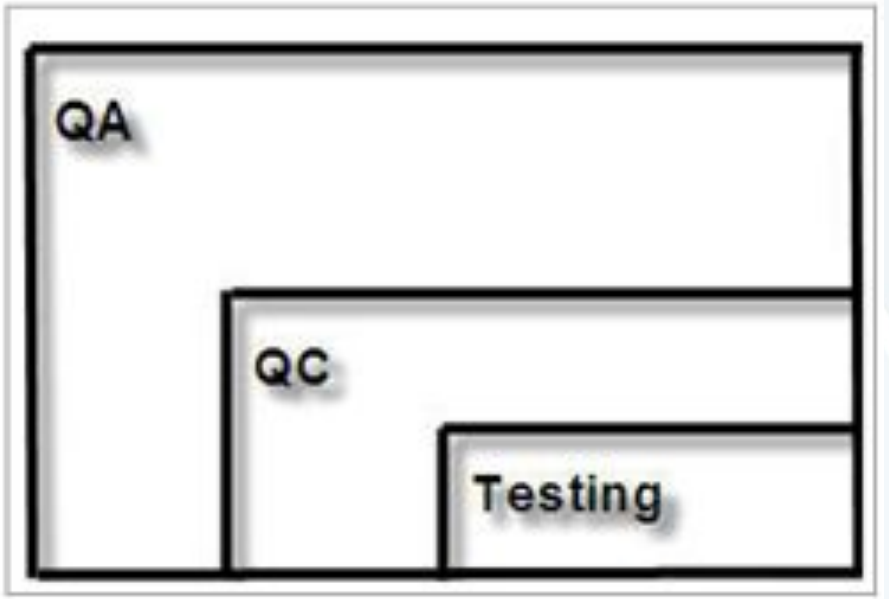
- Проверка требований к продукту;
- Оценка рисков;
- Планирование идей по улучшению качества продукта;
- Планирование тестирования;
- Анализ результатов тестирования.



Внутри процесса QA выделяют процесс Quality Control – контроль качества продукта. QC-специалисты анализируют результаты тестирования и отвечают за выявление и уничтожение дефектов в продукте.

Еще более узкая специальность в рамках QA/QC – тестировщик ПО, который проверяет готовый продукт на наличие ошибок (багов) и несоответствие требованиям, и затем документирует найденные дефекты и пути их воспроизведения. Тестирование – это один из этапов обеспечения и контроля качества.

В Украине различия между должностями QA и тестировщика смазаны, и на практике это одно и то же. Хотя теоретически тестировщик тестирует продукт как результат, а QA работает над обеспечением процессов, которые могут повысить качество ПО в целом.



A-Level UKRAINE

Выделяют 4 основные роли:

Test Analyst – занимается статическим тестированием требований: проверяет, насколько они полны, однозначны, непротиворечивы и т. д.

Test Designer – создает набор тестов на базе требований, планирует конфигурации, необходимые для тестирования.

Test Executor – выполняет заранее подготовленные тесты, документирует найденные ошибки и шаги их воспроизведения.

Test Manager – скорее управленец, чем инженер. Планирует и контролирует работы, связанные с тестированием: оценки сроков, работу над планом-графиком, контроль покрытия требований тестами, постановку задач членам команды, коммуникацию со стейкхолдерами.

Карьера в IT: должность QA engineer

Automation QA engineer

AQA engineer – это специалист по обеспечению качества продукта, который использует программные средства для создания тестов и проверки результатов выполнения.

Основная задача AQA – создавать автоматические скрипты, которые будут проверять работу программы на основании тест-кейсов, написанных manual QA. Это помогает сократить время тестирования и упростить его процесс.

AQA обладает навыками программиста и логикой тестировщика одновременно:

— Как и QA-инженеры или тестировщики, AQA мониторит качество продукта на различных этапах его разработки, тестирования и эксплуатации.

— Как и программисты, AQA занимается разработкой, только он создает продукт, чтобы проверить написанное программистами.

Другими словами, программисты – создают, тестировщики – ломают, а автоматизаторы – создают, чтобы сломать.

[Карьера в IT: должность QA Automation engineer]

Жизненный цикл программного обеспечения

Жизненный цикл программного обеспечения (ПО) – период времени, состоящий из нескольких этапов, который начинается с момента принятия решения о необходимости создания программного продукта и заканчивается в момент его полного изъятия из эксплуатации. Этапы могут называться по-разному и дробиться на более мелкие стадии.

Рассмотрим эти этапы на примере жизненного цикла интернет-магазина.

Подготовка. Иван решил запустить книжный интернет-магазин и начал анализировать, какие подобные сайты уже представлены в сети. Собрал информацию об их трафике, функциональности.

Проектирование. Иван выбрал компанию-подрядчика и обсудил с её специалистами архитектуру и дизайн будущего интернет-магазина.

Создание. Иван заключил с разработчиками договор. Они начали писать код, отрисовывать дизайн, составлять документацию.

Поддержка. Иван подписал акт сдачи-приёмки, и подрядчик разместил интернет-магазин на «боевых» серверах. Пользователи начали его посещать и сообщать о замеченных ошибках в поддержку, а программисты — оперативно всё исправлять.

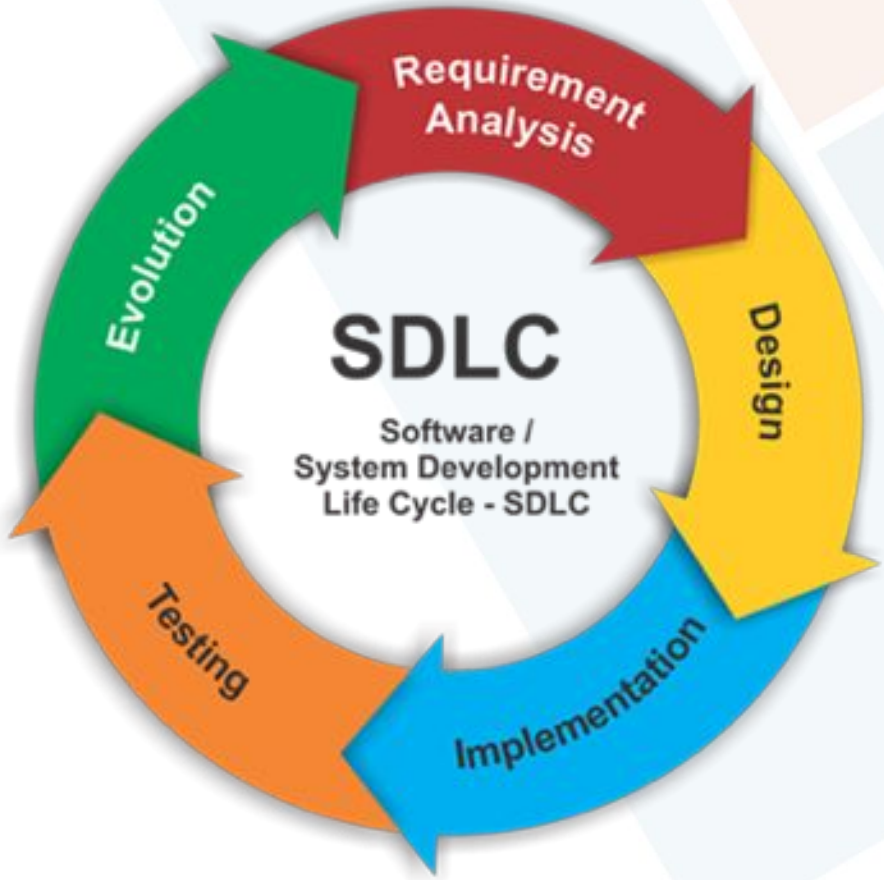
A-Level UKRAINE

Модель жизненного цикла ПО

Модель жизненного цикла ПО – структура, определяющая последовательность выполнения и взаимосвязи процессов, действий и задач на протяжении жизненного цикла. Модель жизненного цикла зависит от специфики, масштаба и сложности проекта и специфики условий, в которых система создается и функционирует.

Модель разработки программного обеспечения описывает, какие стадии жизненного цикла оно проходит и что происходит на каждой из них.

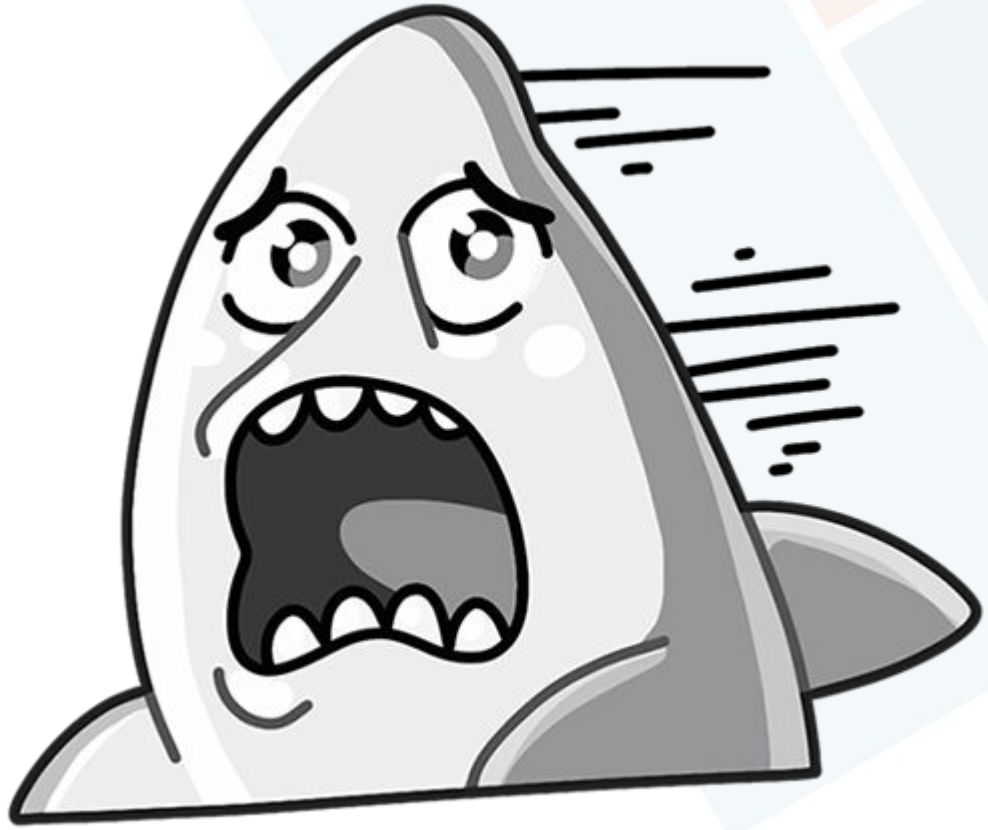
А методология включает в себя набор методов по управлению разработкой: это правила, техники и принципы, которые делают её более эффективной.



A-Level UKRAINE

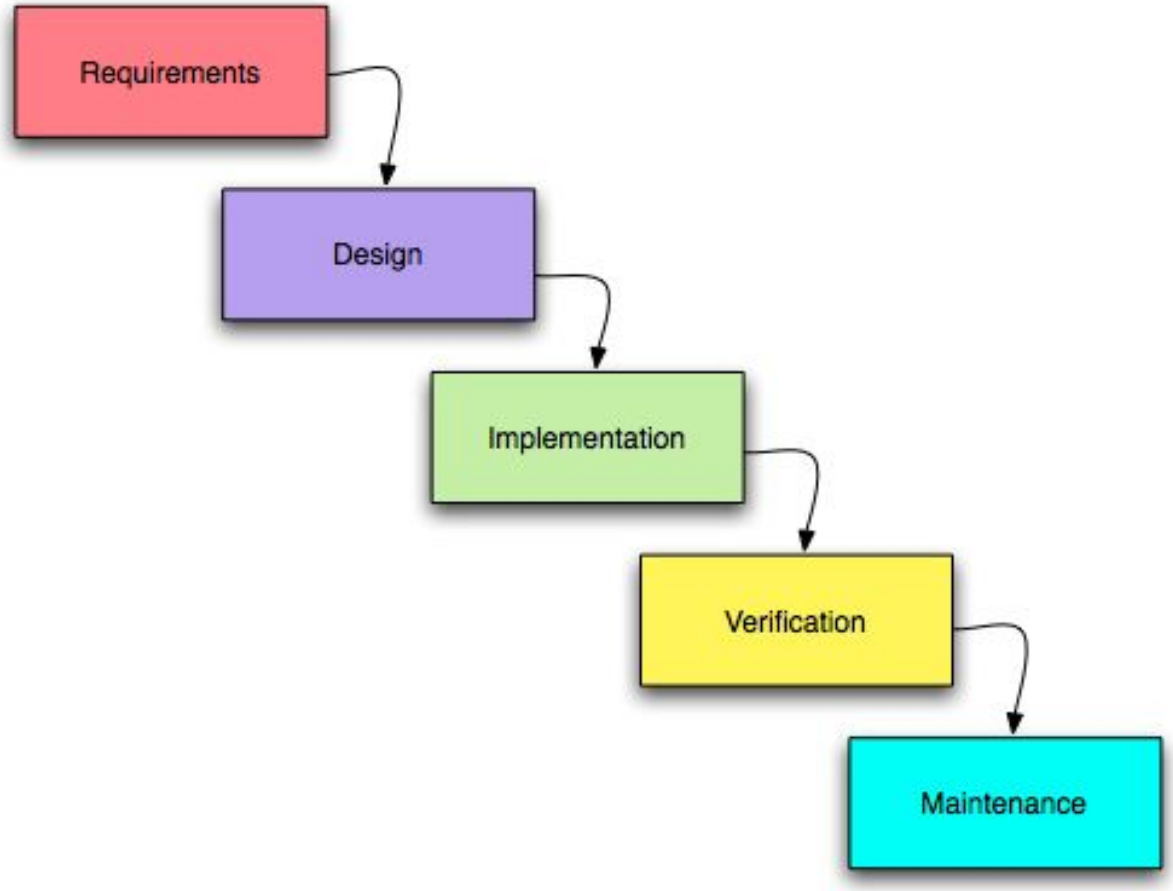
Основные модели разработки ПО

- Waterfall (каскадная модель или «водопад»);
- V-model – V-образная модель, разработка через тестирование;
- Incremental Model – инкрементная модель;
- Iterative Model – итеративная (или итерационная) модель;
- Spiral Model – спиральная модель;



A-Level UKRAINE

Каскадная модель (waterfall)



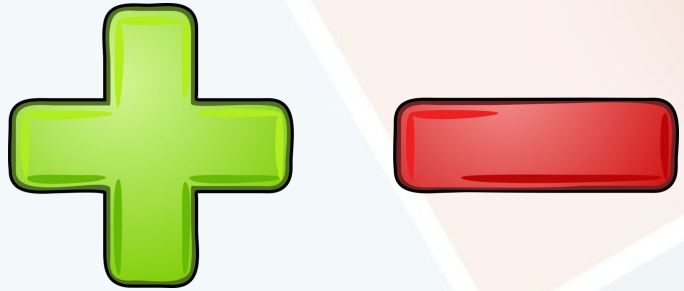
Основная суть модели Waterfall в том, что этапы зависят друг от друга и следующий начинается, когда закончен предыдущий, образуя таким образом поступательное (каскадное) движение вперед.

Когда использовать каскадную методологию?

- Только тогда, когда требования известны, понятны и зафиксированы;
- Противоречивых требований не имеется;
- Нет проблем с доступностью программистов нужной квалификации;
- В относительно небольших проектах.

«Водопад» подходит для разработки проектов в медицинской и космической отрасли, где уже сформирована обширная база документов, на основе которых можно написать требования к новому ПО.

При работе с каскадной моделью основная задача – написать подробные требования к разработке. На этапе тестирования не должно выясниться, что в них есть ошибка, которая влияет на весь продукт.



Преимущества каскадной модели:

- Разработку просто контролировать. Заказчик всегда знает, чем сейчас заняты программисты, может управлять сроками и стоимостью.
- Стоимость проекта определяется на начальном этапе. Все шаги запланированы уже на этапе согласования договора, ПО пишется непрерывно «от и до».
- Не нужно нанимать тестировщиков с серьёзной технической подготовкой. Тестировщики смогут опираться на подробную техническую документацию.

Недостатки каскадной модели:

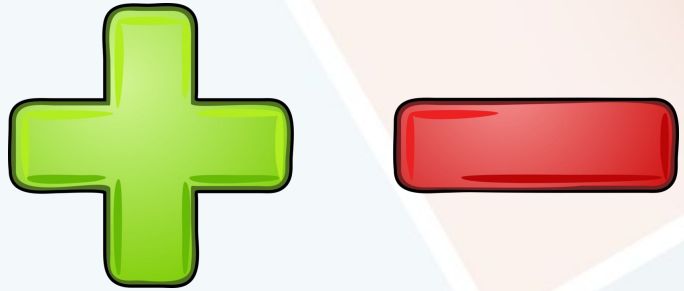
- Тестирование начинается на последних этапах разработки. Если в требованиях к продукту была допущена ошибка, то исправить её будет стоить дорого. Тестировщики обнаружат её, когда разработчик уже написал код, а технические писатели — документацию.
- Заказчик видит готовый продукт в конце разработки и только тогда может дать обратную связь. Велика вероятность, что результат его не устроит.
- Разработчики пишут много технической документации, что задерживает работы. Чем обширнее документация у проекта, тем больше изменений нужно вносить и дольше их согласовывать.

V-модель

Это усовершенствованная каскадная модель, в которой заказчик с командой разработки одновременно составляют требования к системе и описывают, как будут тестировать её на каждом этапе. История этой модели начинается в 1980-х.

V-модель подходит для проектов, в которых важна надёжность и цена ошибки очень высока. Например, при разработке подушек безопасности для автомобилей или систем наблюдения за пациентами в клиниках.

[Difference between V-model and W-model in software testing.]



Преимущества V-модели:

- Каждая стадия имеет конкретные результаты;
- Более высокие показатели по сравнению с каскадной моделью по причине того, что тестирование начинается на ранних этапах;
- Экономия времени по сравнению с каскадной моделью может достигать 50%;
- Отлично подходит для небольших проектов, где все требования к продукту очевидны сразу;
- Полноценная реализация доступных ресурсов.

Недостатки V-модели:

- Отсутствие гибкости, как и в случае с каскадной моделью. Вносить изменения на поздних этапах будет трудно и дорого
- Сама разработка начинается строго с началом соответствующей стадии, то есть, никаких прототипов на ранних этапах не разрабатывается
- Контроль рисков затруднен: нет определённого способа решения критических проблем, обнаруженных на этапе тестирования

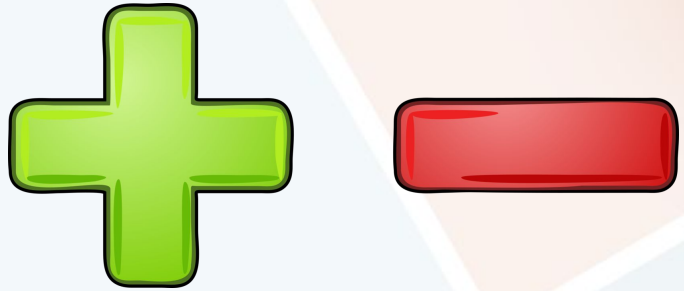
A-Level UKRAINE

Итерационная модель

Итерационная модель предполагает разбиение проекта на части (этапы, итерации) и прохождение этапов жизненного цикла на каждом их них. Каждый этап является законченным сам по себе, совокупность этапов формирует конечный результат.

Это модель, при которой заказчик не обязан понимать, какой продукт хочет получить в итоге, и может не прописывать сразу подробные требования.

A-Level UKRAINE



Преимущества итерационной модели:

- Быстрый выпуск минимального продукта даёт возможность оперативно получать обратную связь от заказчика и пользователей. А значит, фокусироваться на наиболее важных функциях ПО и улучшать их в соответствии с требованиями рынка и пожеланиями клиента.
- Постоянное тестирование пользователями позволяет быстро обнаруживать и устранять ошибки.

Недостатки итерационной модели:

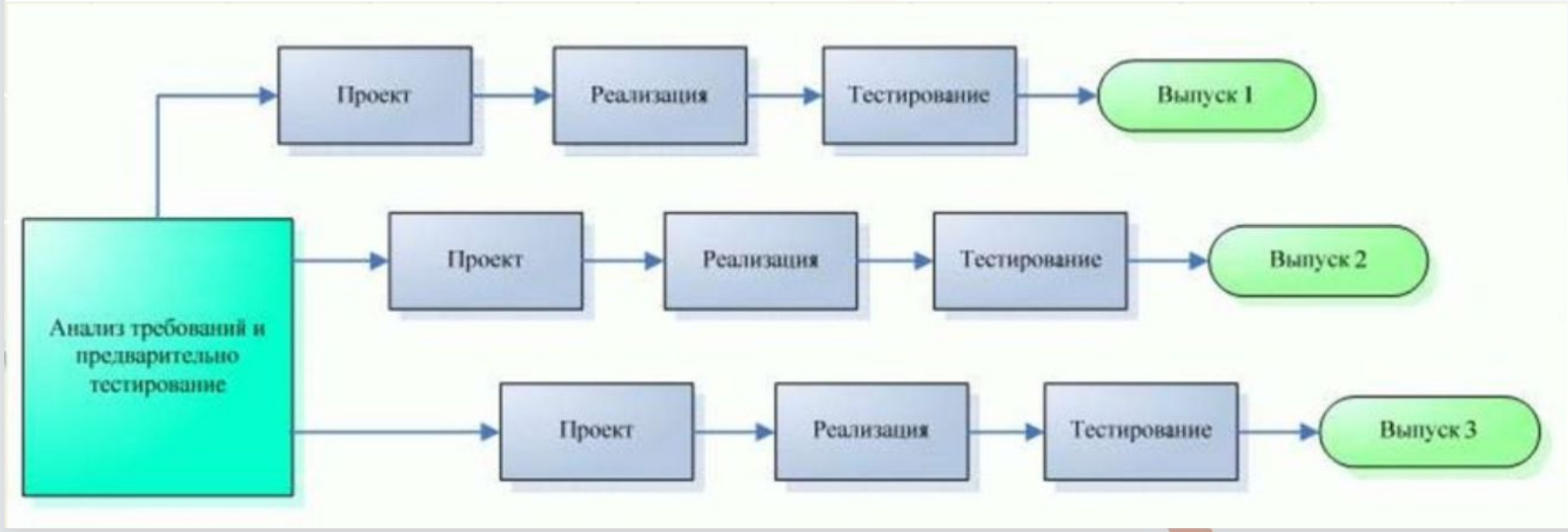
- Использование на начальном этапе баз данных или серверов — первые сложно масштабировать, а вторые не выдерживают нагрузку. Возможно, придётся переписывать большую часть приложения.
- Отсутствие фиксированного бюджета и сроков. Заказчик не знает, как выглядит конечная цель и когда закончится разработка.

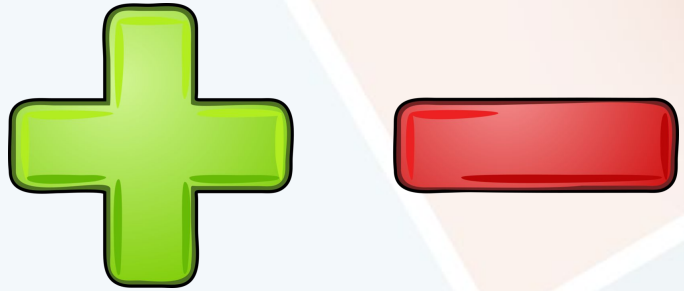
Инкрементная модель

Принцип, который лежит в основе инкрементной модели, подразумевает расширение возможностей, достраивание модулей и функций приложения.

Буквальный перевод слова инкремент: «увеличение на один». Это «увеличение на один» применяется в том числе для обозначения версий продукта.

Если в каскадной модели по сути есть два состояния продукта: «ничего» и «готовый продукт», то с появлением итерационных моделей стало применяться версионирование продукта.





Преимущества инкрементной модели:

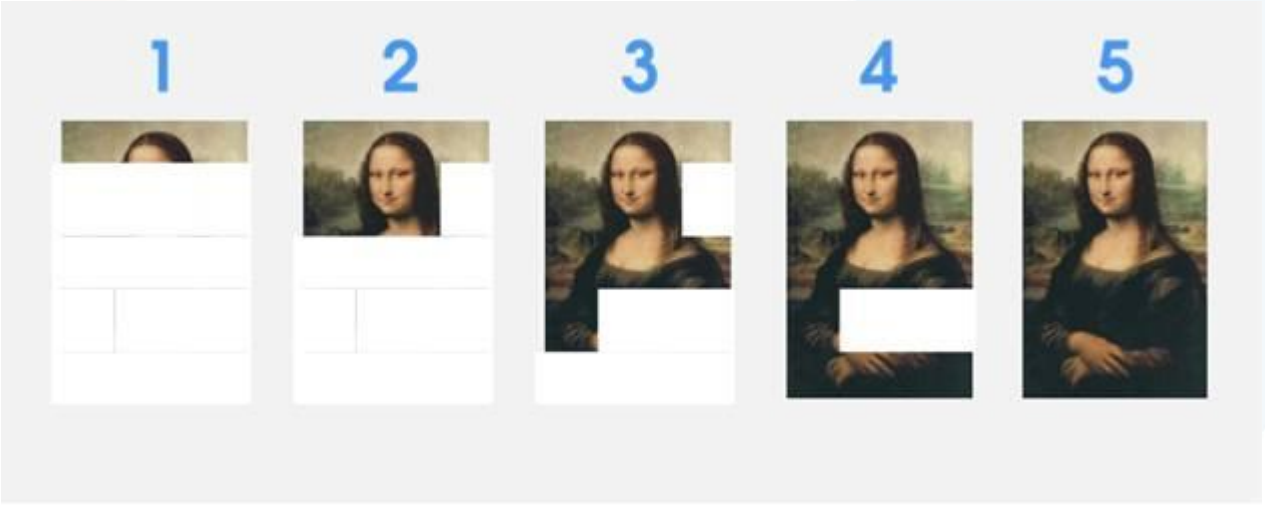
- Не нужно вкладывать много денег на начальном этапе. Заказчик оплачивает создание основных функций, получает продукт, «выкатывает» его на рынок – и по итогам обратной связи решает, продолжать ли разработку.
- Можно быстро получить фидбэк от пользователей и оперативно обновить техническое задание. Так снижается риск создать продукт, который никому не нужен.
- Ошибка обходится дешевле. Если при разработке архитектуры была допущена ошибка, то исправить её будет стоить не так дорого, как в «водопаде» или V-образной модели.

Недостатки инкрементной модели:

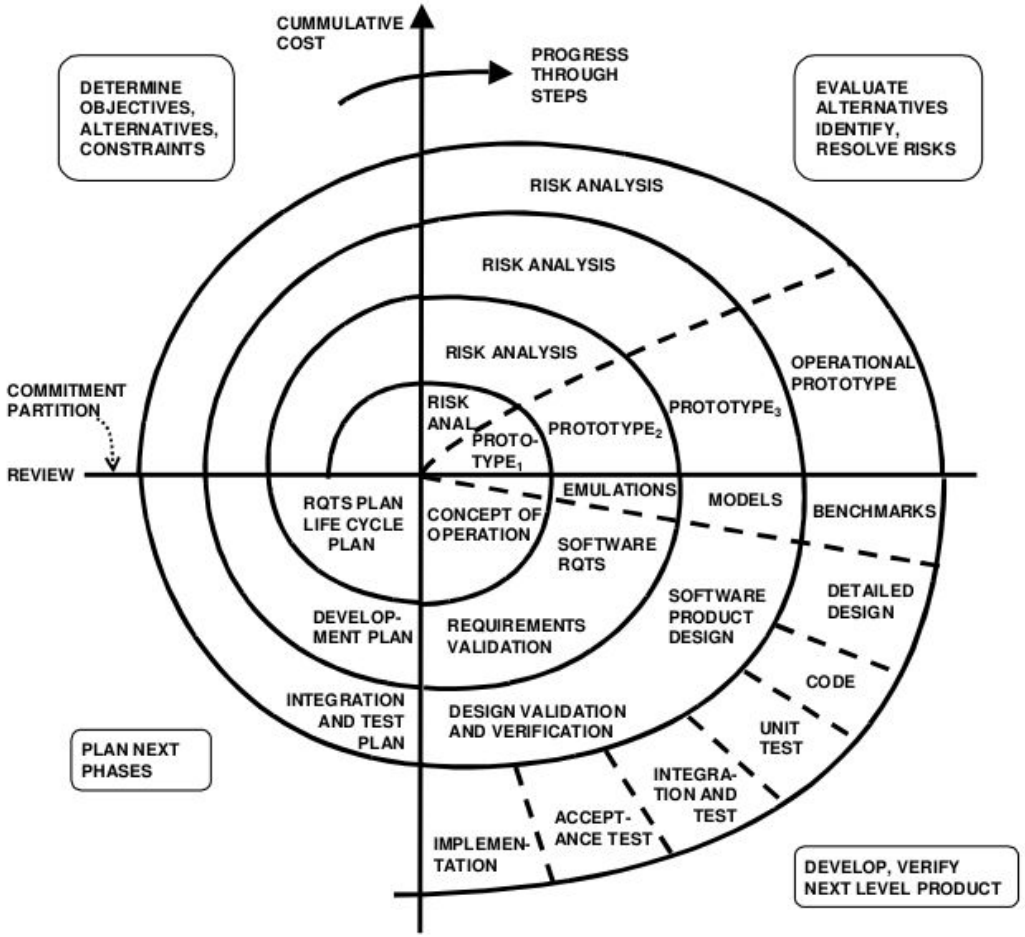
- Отсутствие гибкости, как и в случае с каскадной моделью. Вносить изменения на поздних этапах будет трудно и дорого.
- Сама разработка начинается строго с началом соответствующей стадии, то есть, никаких прототипов на ранних этапах не разрабатывается.
- Контроль рисков затруднен: нет определённого способа решения критических проблем, обнаруженных на этапе тестирования.

Итерационная модель

Инкрементная модель



Спиральная модель



Спиральная модель

Отличительной особенностью этой модели является специальное внимание рискам, влияющим на организацию жизненного цикла.

Каждый виток спирали соответствует созданию фрагмента или версии программного обеспечения, на нём уточняются цели и характеристики проекта, определяется его качество и планируются работы следующего витка спирали. Таким образом углубляются и последовательно конкретизируются детали проекта и в результате выбирается обоснованный вариант, который доводится до реализации. Каждый виток разбит на 4 сектора:

- определение целей;
- оценка и разрешение рисков;
- разработка и тестирование;
- планирование следующей итерации.

A-Level UKRAINE

Гибкая методология разработки

Гибкая методология разработки (англ. agile software development) – обобщающий термин для целого ряда подходов и практик, основанных на ценностях Манифеста гибкой разработки программного обеспечения (Agile Manifesto) и 12 принципах, лежащих в его основе.

Ценности Agile Manifesto

A-Level UKRAINE

Основополагающие принципы Agile Manifesto:

- Наивысшим приоритетом для нас является удовлетворение потребностей заказчика, благодаря регулярной и ранней поставке ценного программного обеспечения.
- Изменение требований приветствуется, даже на поздних стадиях разработки. Agile-процессы позволяют использовать изменения для обеспечения заказчику конкурентного преимущества.
- Работающий продукт следует выпускать как можно чаще, с периодичностью от пары недель до пары месяцев.
- На протяжении всего проекта разработчики и представители бизнеса должны ежедневно работать вместе.
- Над проектом должны работать мотивированные профессионалы. Чтобы работа была сделана, создайте условия, обеспечьте поддержку и полностью доверьтесь им.
- Непосредственное общение является наиболее практичным и эффективным способом обмена информацией как с самой командой, так и внутри команды.

- Работающий продукт — основной показатель прогресса.
- Инвесторы, разработчики и пользователи должны иметь возможность поддерживать постоянный ритм бесконечно. Agile помогает наладить такой устойчивый процесс разработки.
- Постоянное внимание к техническому совершенству и качеству проектирования повышает гибкость проекта.
- Простота — искусство минимизации лишней работы — крайне необходима.
- Самые лучшие требования, архитектурные и технические решения рождаются у самоорганизующихся команд.
- Команда должна систематически анализировать возможные способы улучшения эффективности и соответственно корректировать стиль своей работы

Преимущества и недостатки Agile

Главные достоинства Agile – быстрота, адаптивность и фокус на главном. Отсутствие бюрократии и периодичность поставок работающего продукта с постепенным наращиванием его функциональных возможностей существенно сокращают сроки получения итогового результата. Это особенно важно для бизнеса, т.к. благодаря стремительному выходу на рынок можно быстро занять свободную нишу.

Недостатки Agile являются прямым следствием его достоинств:

- снижение важности регламентирующей и технической документации может привести к ее нерелевантности или даже к фактическому отсутствию;
- краткосрочное планирование не всегда учитывает необходимость масштабирования продукта, что влечет ошибки в архитектуре;
- появление новых требований после нескольких итераций приводит к кардинальным изменениям архитектуры и переделкам уже созданных решений;
- накопление дефектов и снижение качества продуктов вследствие решения проблем самым простым и быстрым, но не всегда самым правильным способом.

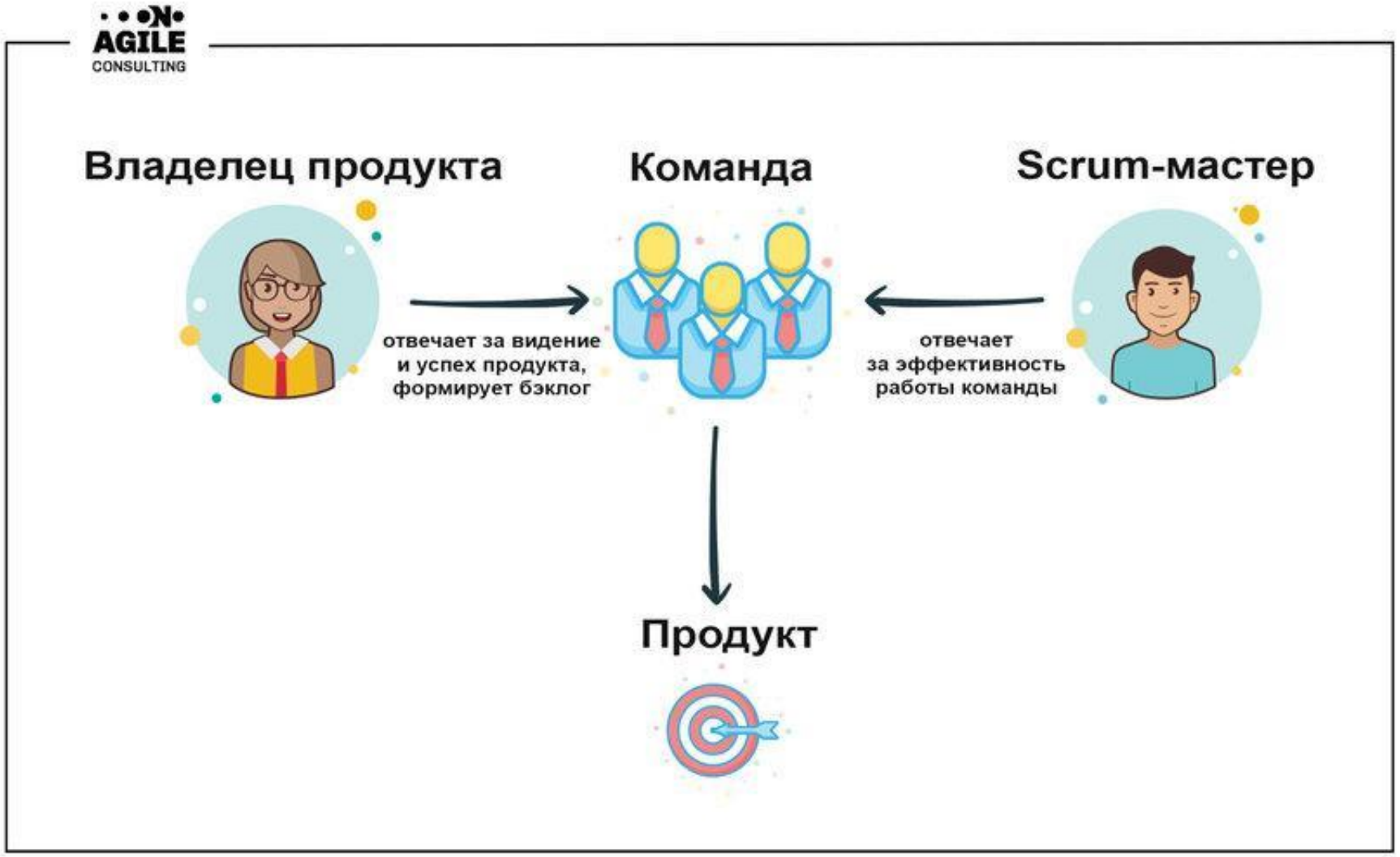
Методология Scrum

Методология Scrum – это набор правил для организации гибкого рабочего процесса, который заключается в командном подходе, работе итерациями, фокусировке на цели каждой итерации и нестандартном распределении обязанностей внутри коллектива.

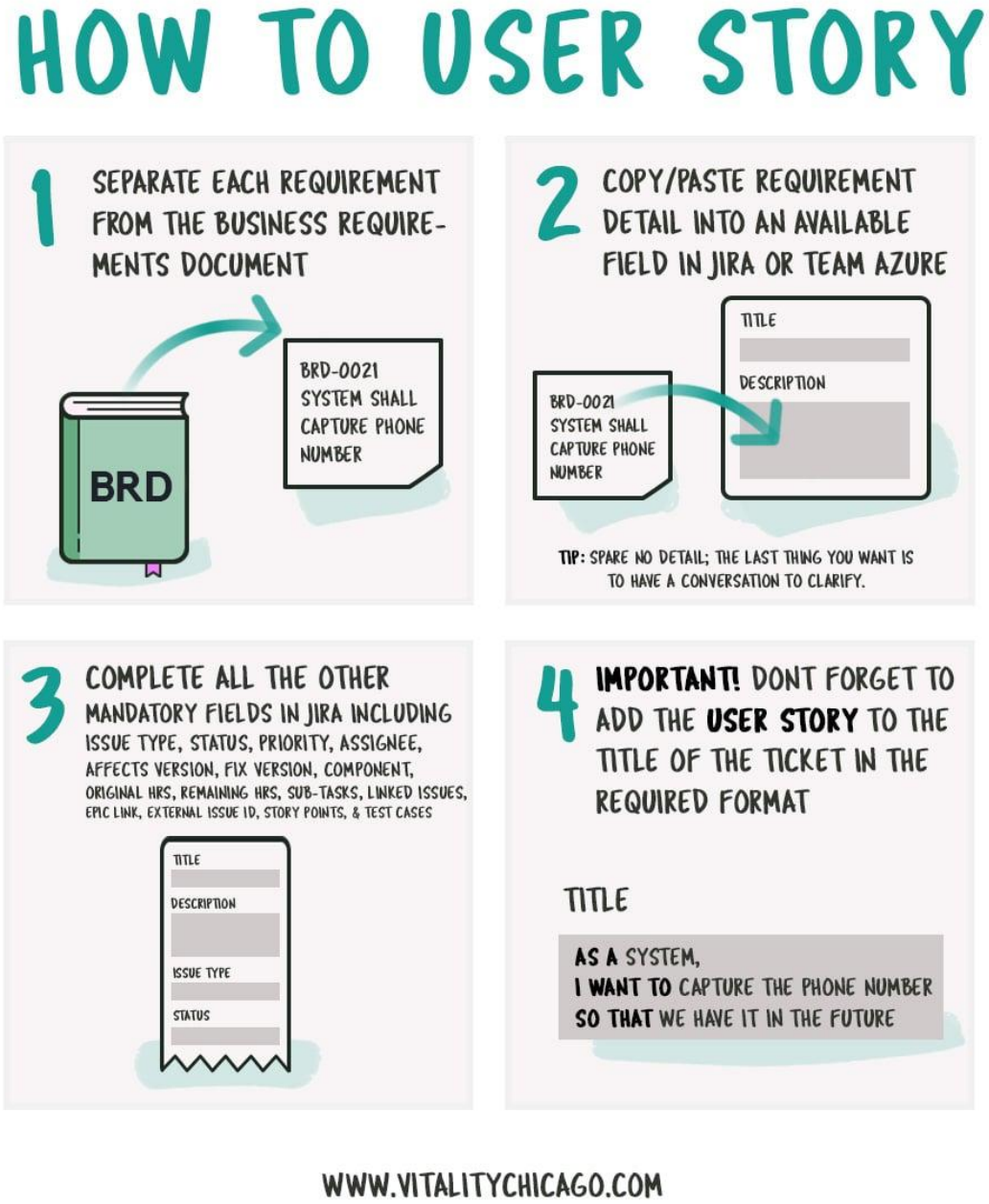
Scrum — итеративный подход. Итерации называются «спринтами», их длительность определяется на старте проекта и фиксирована до конца. Обычно спринты длятся от двух до четырех недель (очень редко — одну неделю). Чем они короче, тем легче работать с изменениями.

В классической методологии Scrum существуют 3 базовых роли:

- Product owner – представляет интересы конечных пользователей и других заинтересованных в продукте сторон.
- Scrum master – роль, которая обеспечивает эффективную реализацию методики Scrum.
- Development team – это люди, выполняющие работу. Сперва можно подумать, что под командой разработчиков подразумеваются только программисты, но это не так. Согласно руководству по Scrum, команда разработчиков может состоять из людей самых разных профессий, в том числе дизайнеров, тестировщиков, программистов и т. д.


AGILE
CONSULTING
Владелец продукта
Команда
Scrum-мастер
отвечает за видение
и успех продукта,
формирует бэклог
отвечает
за эффективность
работы команды
Продукт

User story (пользовательская история) – это короткое, понятное всем описание того, какая функция или свойство требуется продукту.



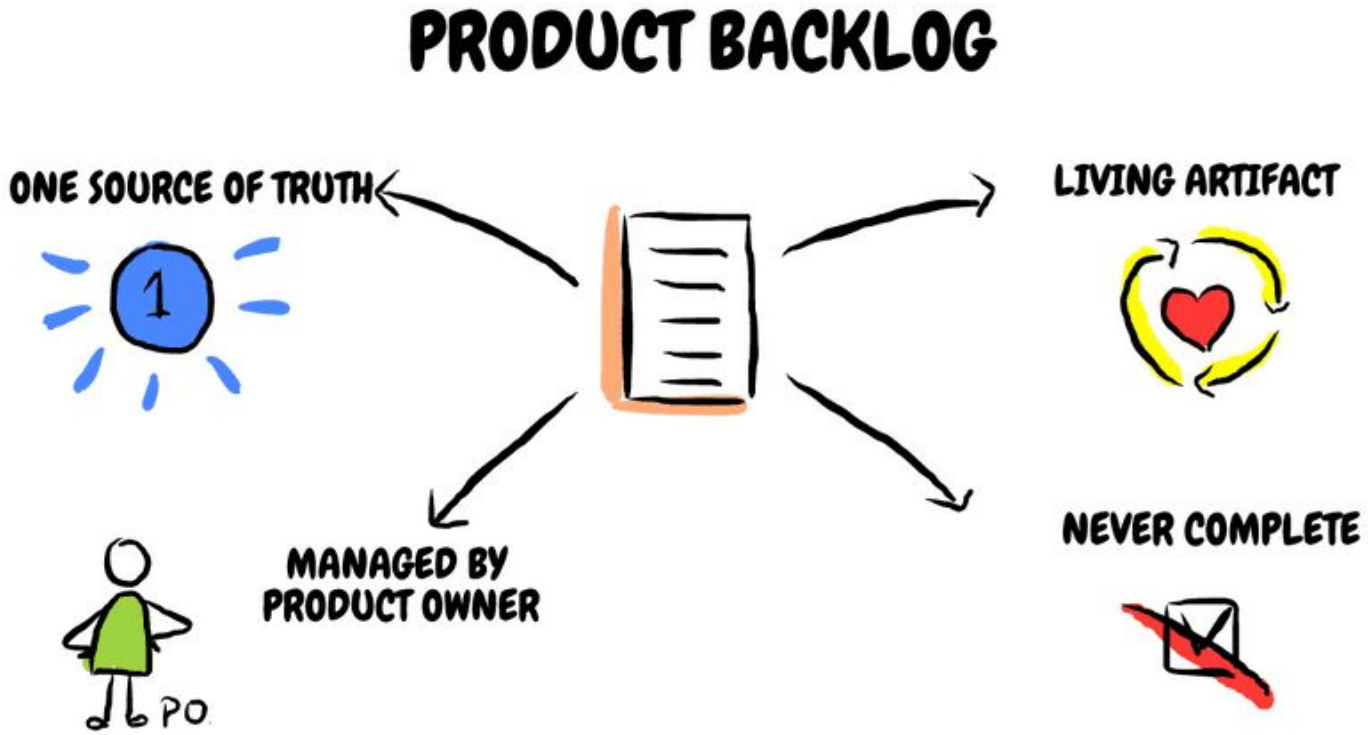
User Story

Title:	Priority:	Estimate:
User Story: As a [description of user], I want [functionality] so that [benefit].		
Acceptance Criteria: Given [how things begin] When [action taken] Then [outcome of taking action]		

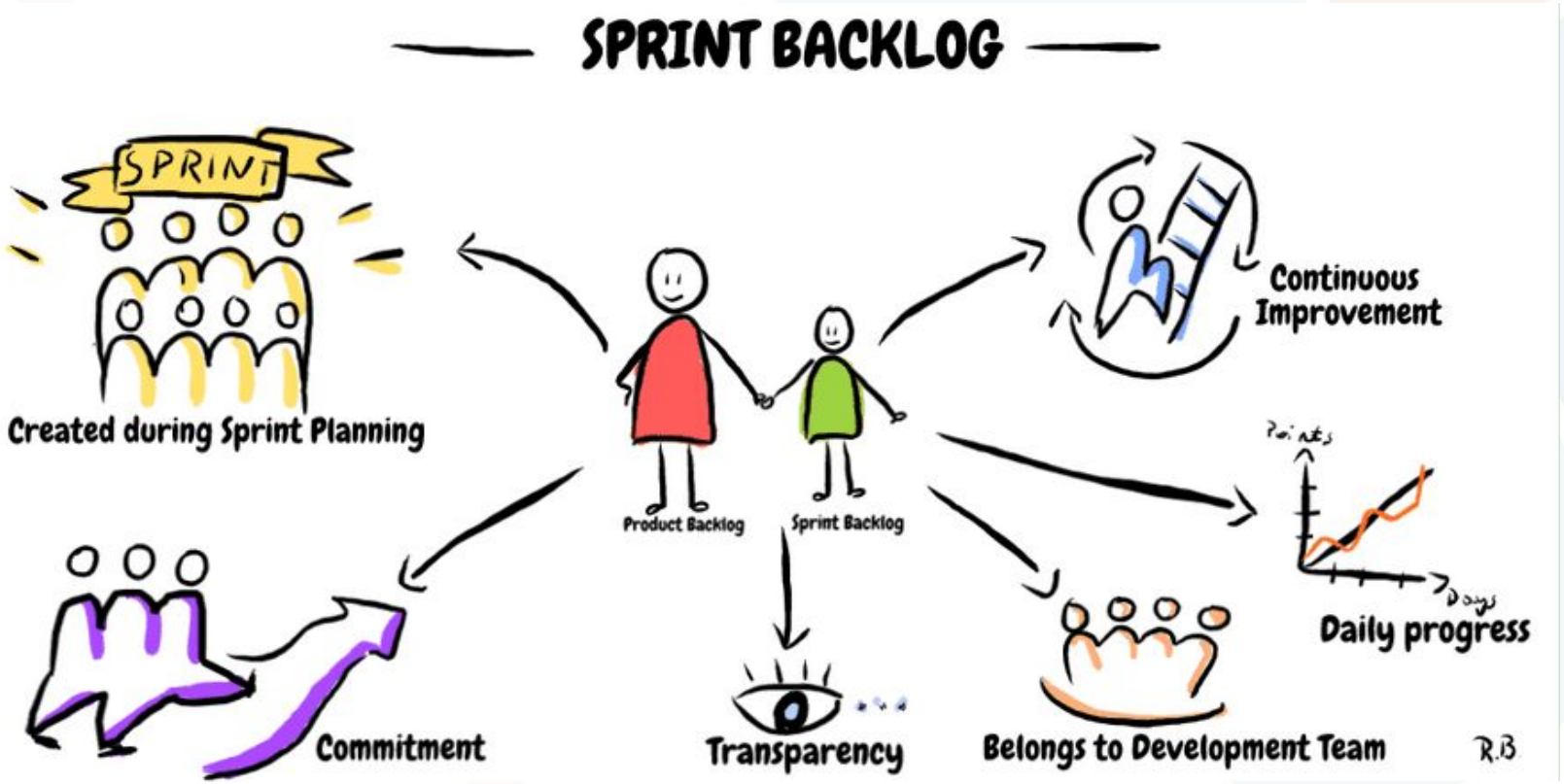
ProductPlan

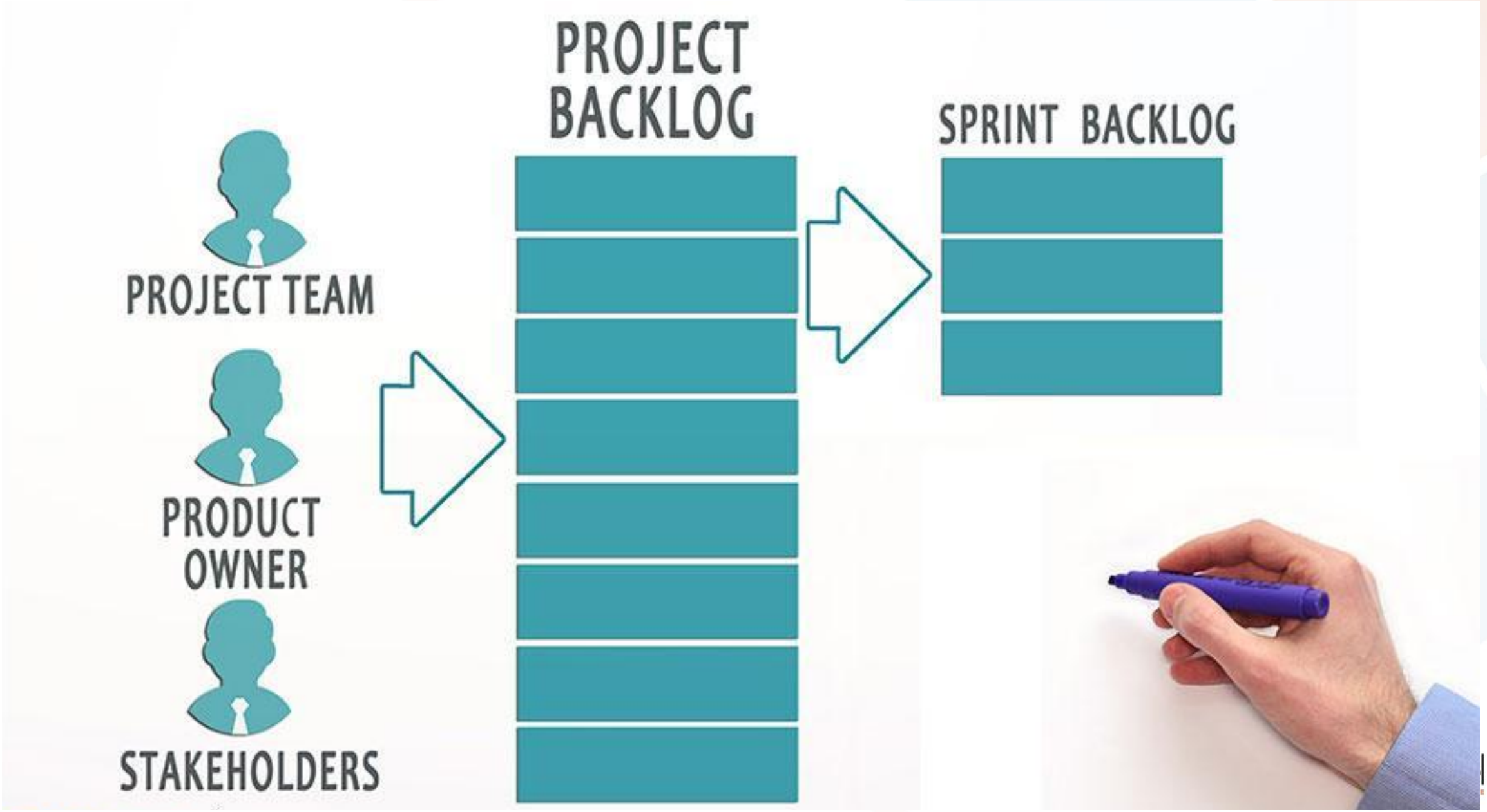
Артефакты Scrum

Product Backlog (также называется общим бэклогом, глобальным бэклогом, или просто бэклогом) – краткий и понятный всем нужным лицам перечень функций и свойств продукта, которые должны быть разработаны. Бэклог продукта состоит из пользовательских историй. Пользовательские истории должны быть отсортированы по приоритетности.

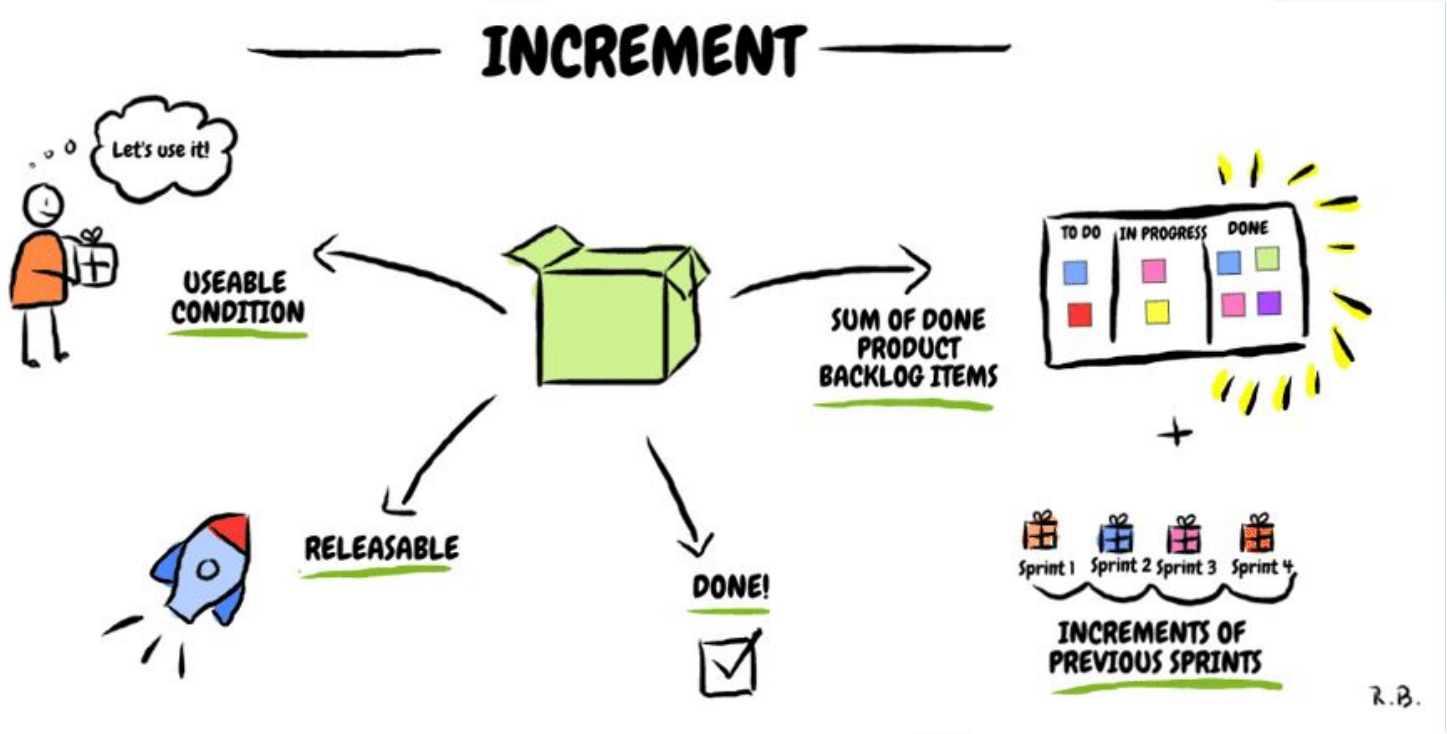


Sprint Backlog – это список задач, которые команда должна выполнить за один спринт.




PROJECT BACKLOG
SPRINT BACKLOG
PROJECT TEAM
PRODUCT OWNER
STAKEHOLDERS

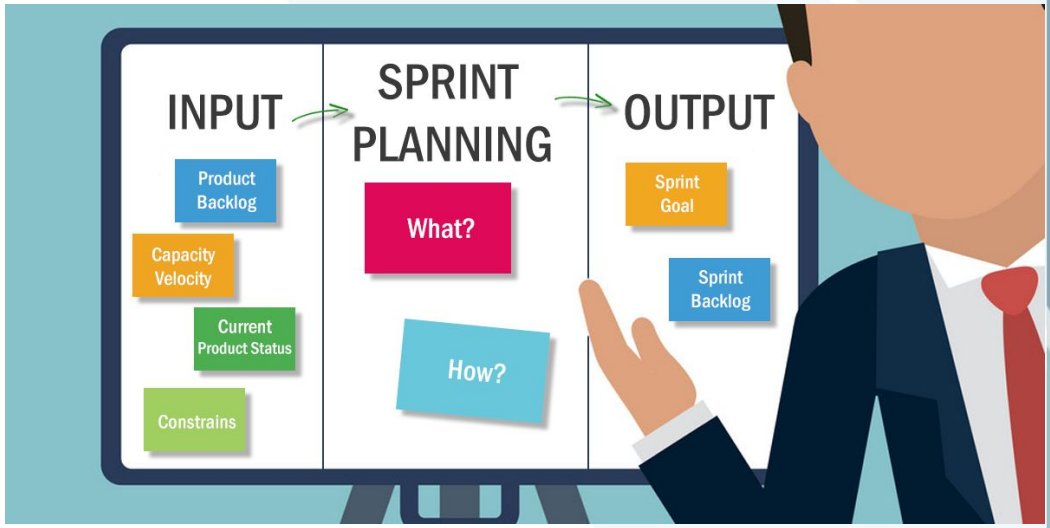
Инкремент продукта представляет собой готовую к использованию часть продукта, которая должна быть реализована к завершению спринта. В конце спринта команда демонстрирует инкремент продукта всем заинтересованным лицам, для получения обратной связи от них и принятия решения по дальнейшему направлению развития продукта.



Scrum meetings

Sprint planning meeting

На этом митинге решается судьба спринта. На нём присутствуют Scrum-мастер, Product Owner и команда разработки. Владелец продукта рассказывает, какой результат хочет видеть в конце спринта. Разработчики выясняют нужные моменты во избежание миллиона вопросов, которые могут появиться в процессе. Команда разбирается с нагрузкой. Исходя из всего этого определяется цель спринта. Так проходит первая часть митинга. Во второй части команда составляет спринт бэклог – задачи, которые нужно реализовать.



Длительность митинга зависит от длительности спринта. На собрание по планированию выделяется по 2 часа на каждую неделю в спринте. Если определяем, что спринты будут длиться 2 недели, то на планирование нужно выделить 4 часа.

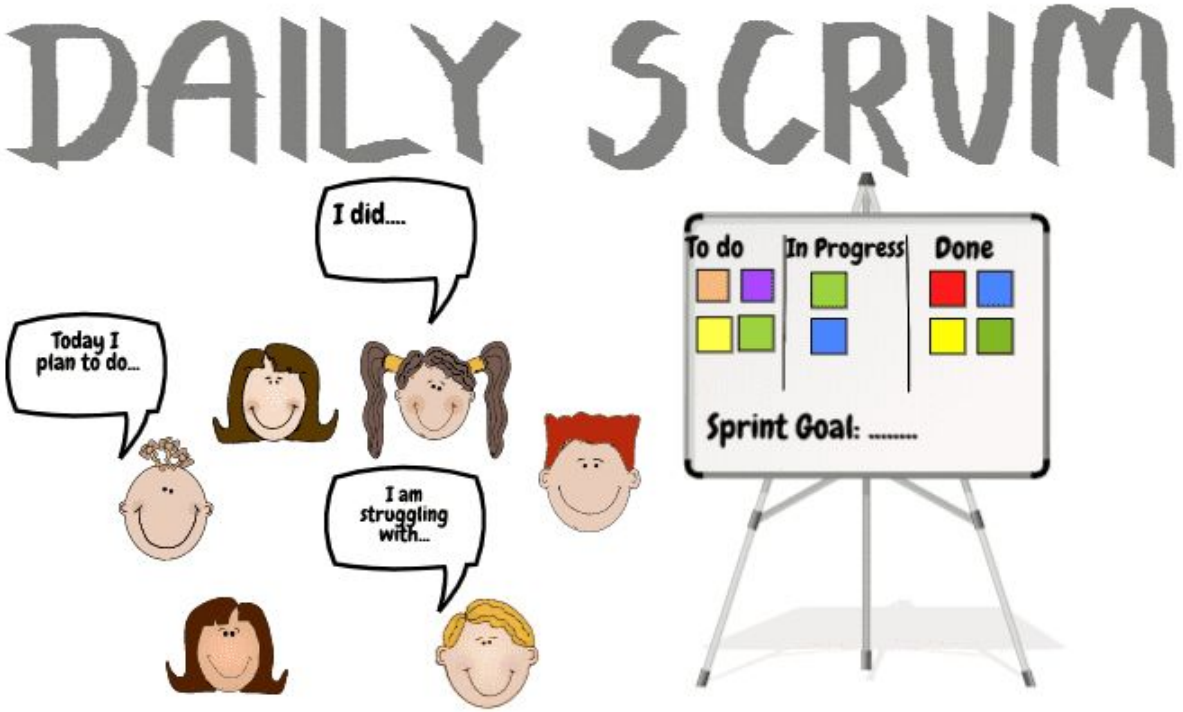
A-Level UKRAINE

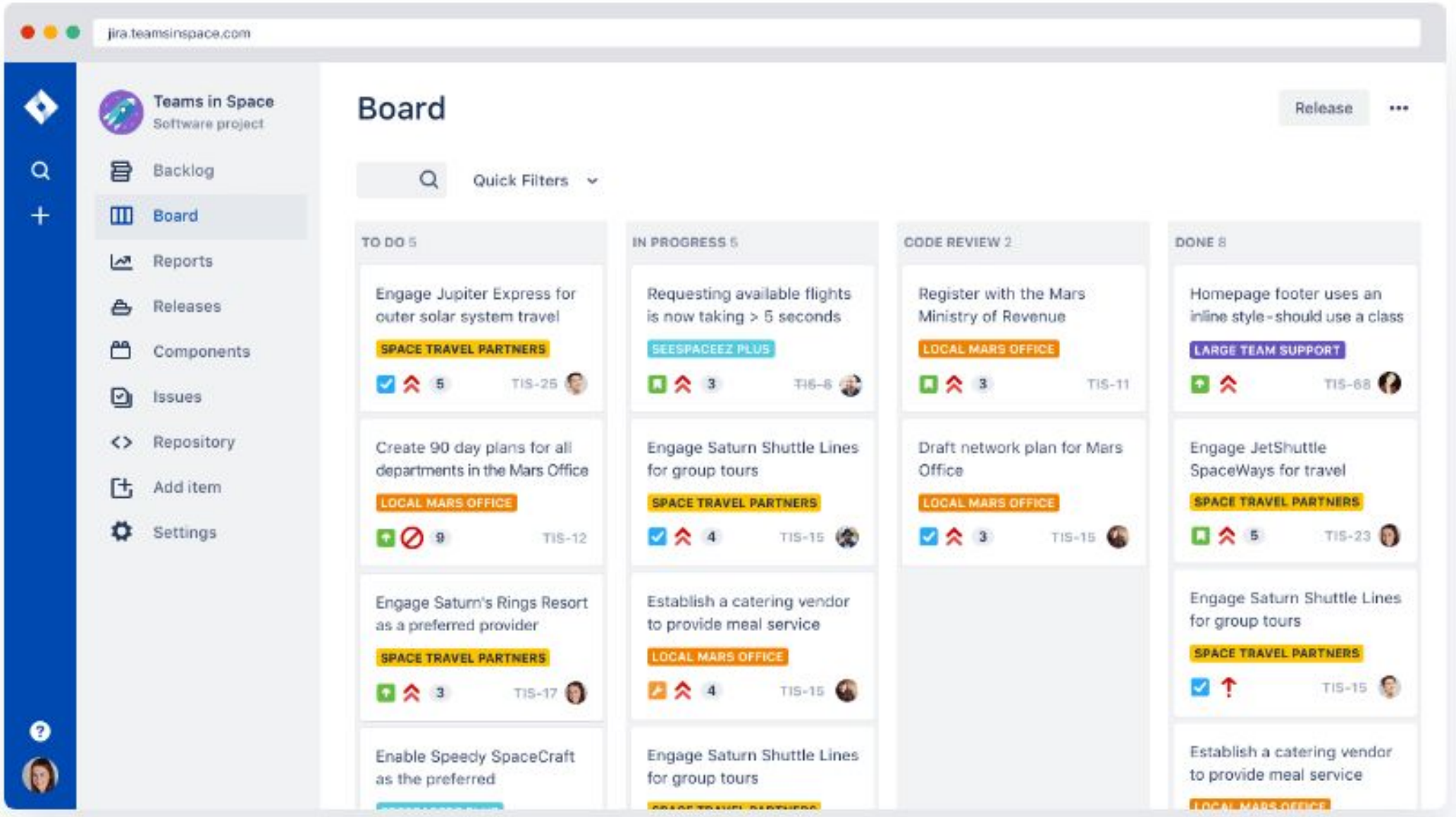
Daily Scrum Meeting

Ежедневное собрание, которое помогает синхронизироваться членам команды. Именно синхронизироваться, а не пожаловаться, похвастаться или обвинить всех в этом мире.

Вопросы, которые хоть и повторяются из митинга в митинг, но которые очень помогают понять текущую ситуацию по прогрессу:

- Что было сделано?
- Что запланировано?
- Есть ли проблемы, и что может помочь?




jira.teamsinspace.com
Teams in Space
Software project
Backlog
Board
Reports
Releases
Components
Issues
Repository
Add item
Settings
Board
Release
Quick Filters
TO DO 5
Engage Jupiter Express for outer solar system travel
SPACE TRAVEL PARTNERS
TIS-25
Create 90 day plans for all departments in the Mars Office
LOCAL MARS OFFICE
TIS-12
Engage Saturn's Rings Resort as a preferred provider
SPACE TRAVEL PARTNERS
TIS-17
Enable Speedy SpaceCraft as the preferred
IN PROGRESS 5
Requesting available flights is now taking > 5 seconds
SEESPACEEZ PLUS
TIS-6
Engage Saturn Shuttle Lines for group tours
SPACE TRAVEL PARTNERS
TIS-15
Establish a catering vendor to provide meal service
LOCAL MARS OFFICE
TIS-15
Engage Saturn Shuttle Lines for group tours
CODE REVIEW 2
Register with the Mars Ministry of Revenue
LOCAL MARS OFFICE
TIS-11
Draft network plan for Mars Office
LOCAL MARS OFFICE
TIS-15
DONE 8
Homepage footer uses an inline style - should use a class
LARGE TEAM SUPPORT
TIS-68
Engage JetShuttle SpaceWays for travel
SPACE TRAVEL PARTNERS
TIS-23
Engage Saturn Shuttle Lines for group tours
SPACE TRAVEL PARTNERS
TIS-15
Establish a catering vendor to provide meal service

Sprint review meeting

Всё, что происходило в спринте, было ради этого. Ревью – двигатель дальнейшего прогресса. Команда разработки наглядно показывает, что было сделано за спринт.

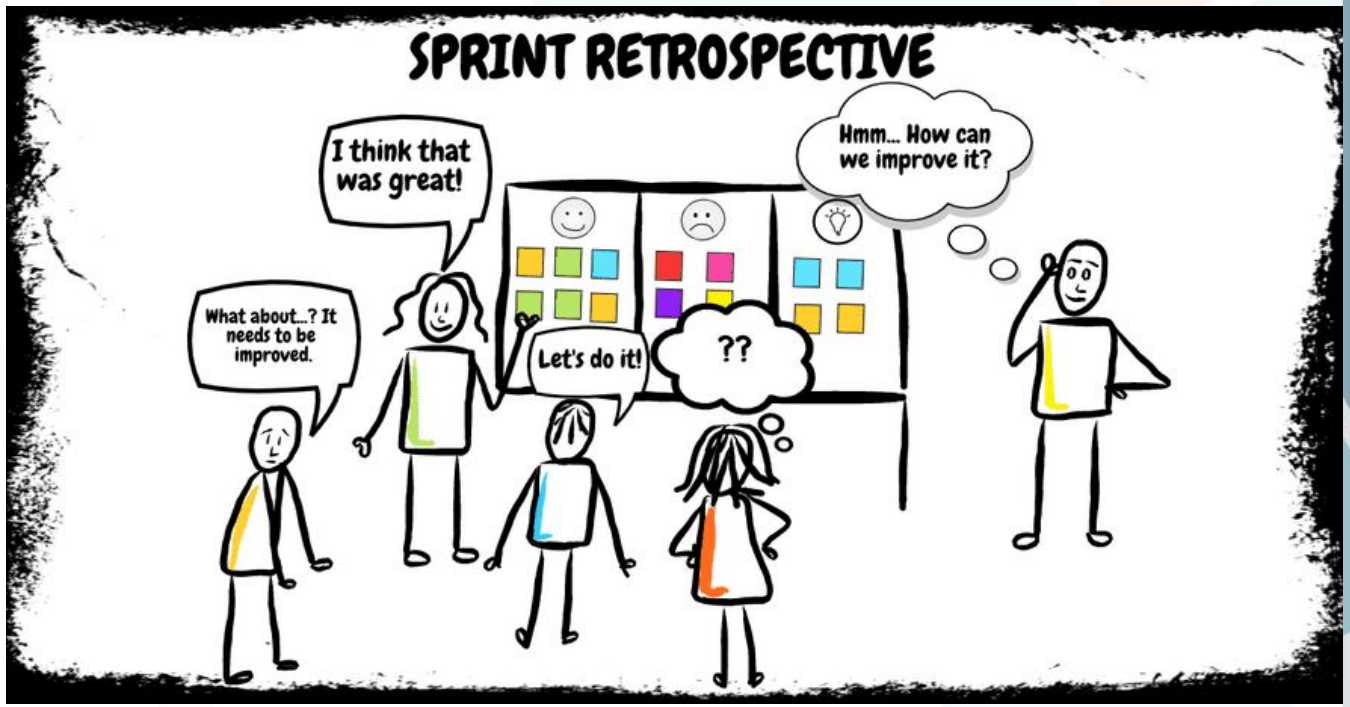
Если показать невозможно, то объясняют, как говорится, на пальцах. Такое бывает, когда сделаны технические задачи, которые продемонстрировать не получится, но без которых ничего или не будет работать, или будет, но плохо. После формируются предложения, мнения, жалобы и видение дальнейшего развития. Всё это берется в расчет на следующий спринт.

Sprint review meeting

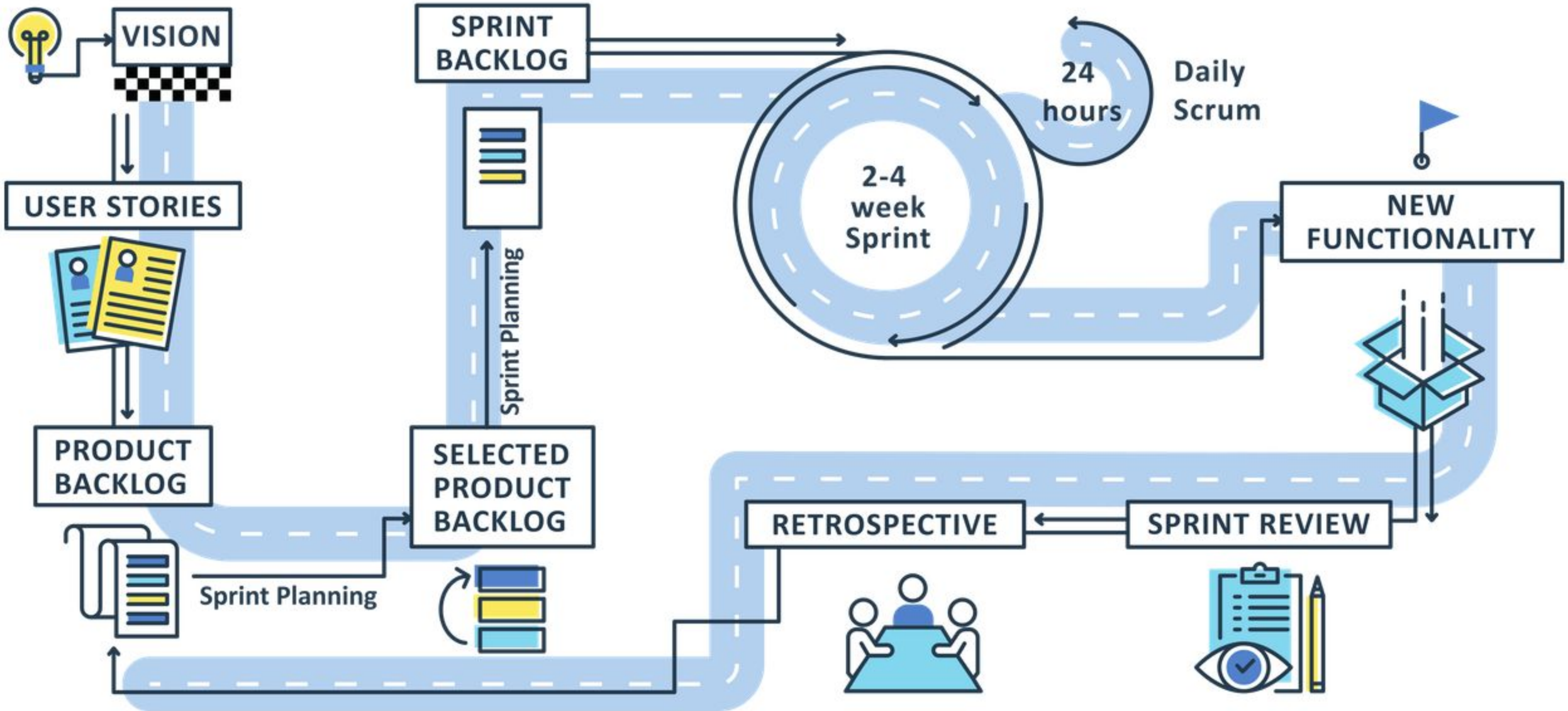
Команда собирается, чтобы выяснить:

- Что было хорошо?
- Что было плохо?
- Что можно улучшить?

Ретро проводится в последний день спринта. Не существует каких-то четких временный рамок. Обычно на этот митинг потребуется 45 минут, умноженные на количество недель в спринте.

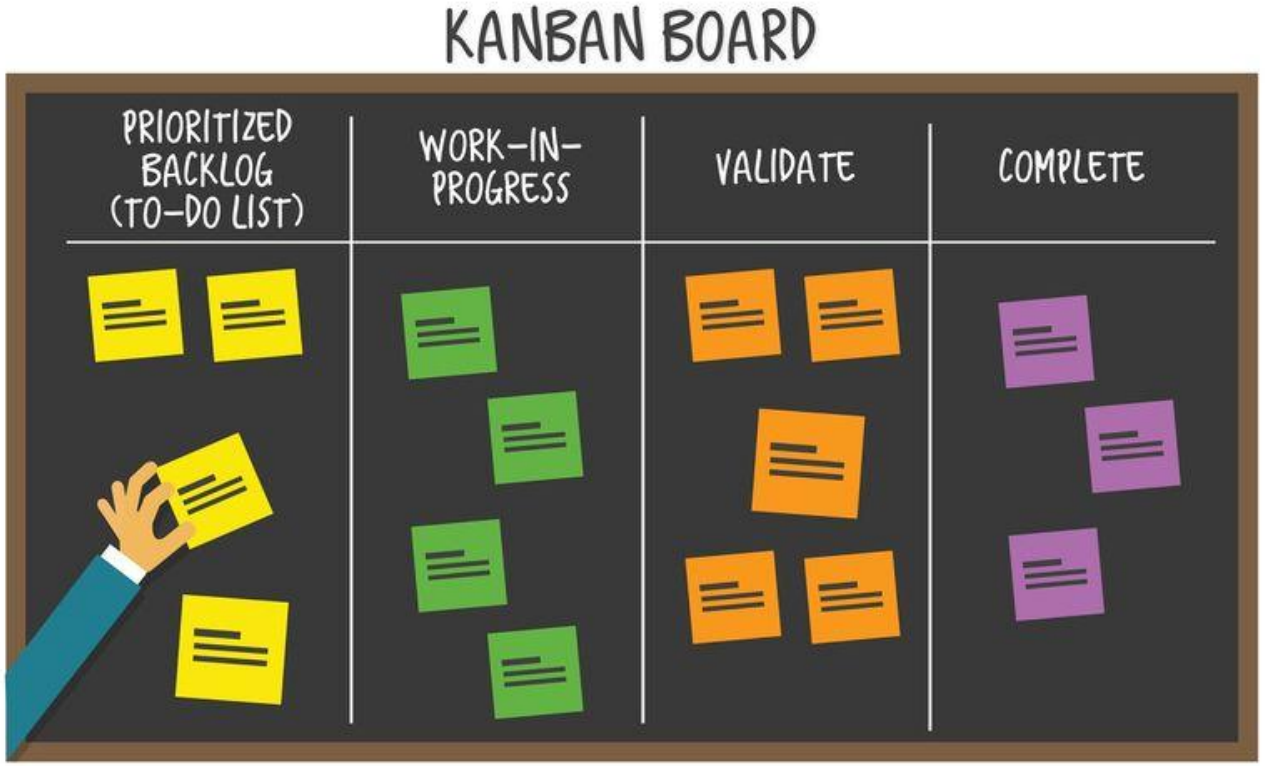


SCRUM PROCESS



Методология Kanban

Методология Kanban – это система постановки задач, при которой все этапы проекта визуализируются на специальной доске. Члены команды могут видеть текущее состояние задачи на любой момент времени. Это предполагает полную прозрачность работы.



Канбан, как и другие практики бережливого производства, пришедшие из Японии, направлен на достижение баланса и выравнивание нагрузки исполнителей. Эффективность работы оценивается по среднему времени жизни задачи, от начальной до конечной стадии. Если задача прошла весь путь быстро, то команда проекта работала продуктивно и слаженно. Иначе – необходимо решать проблему: искать, где и почему возникли задержки и чью работу надо оптимизировать.