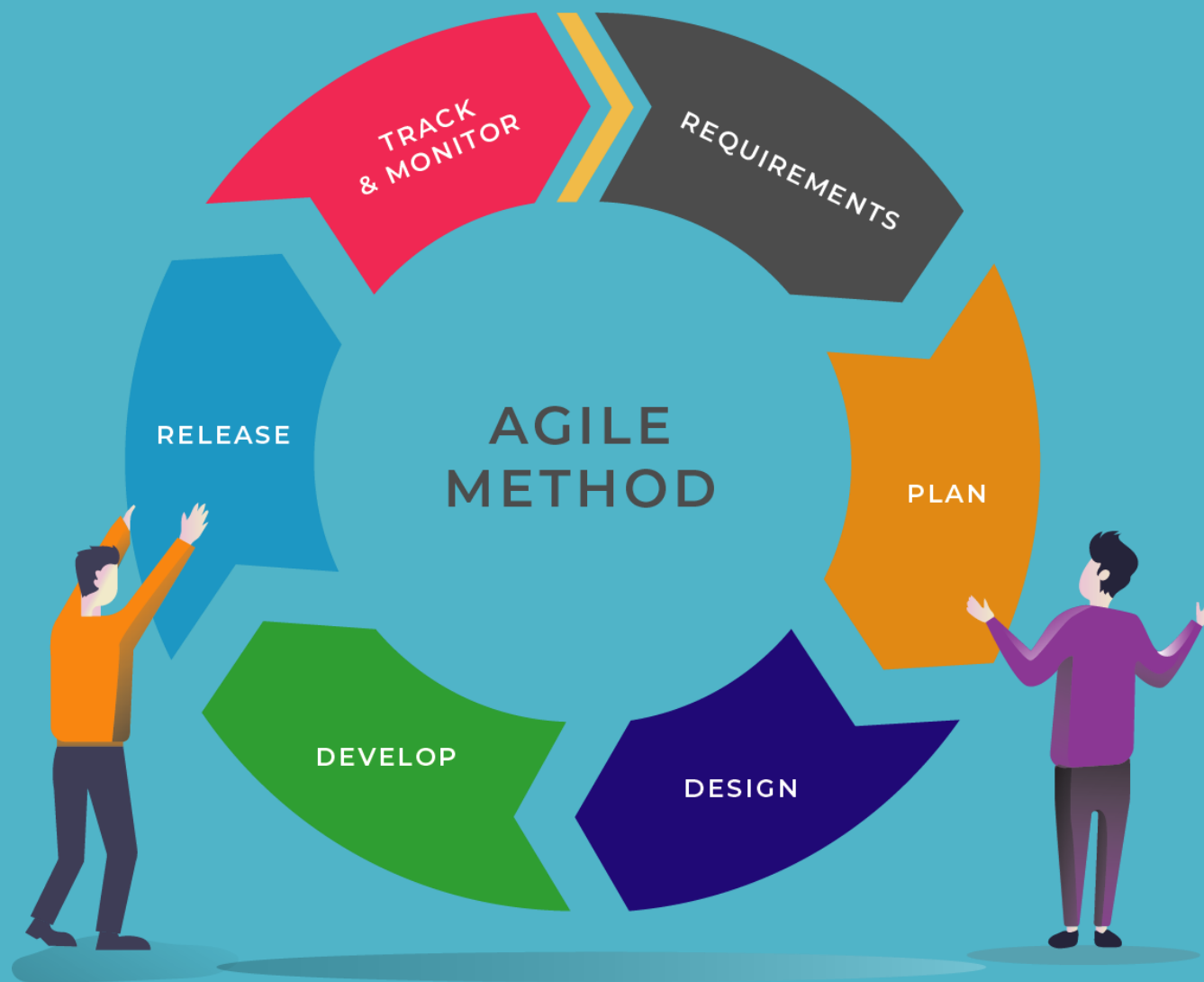
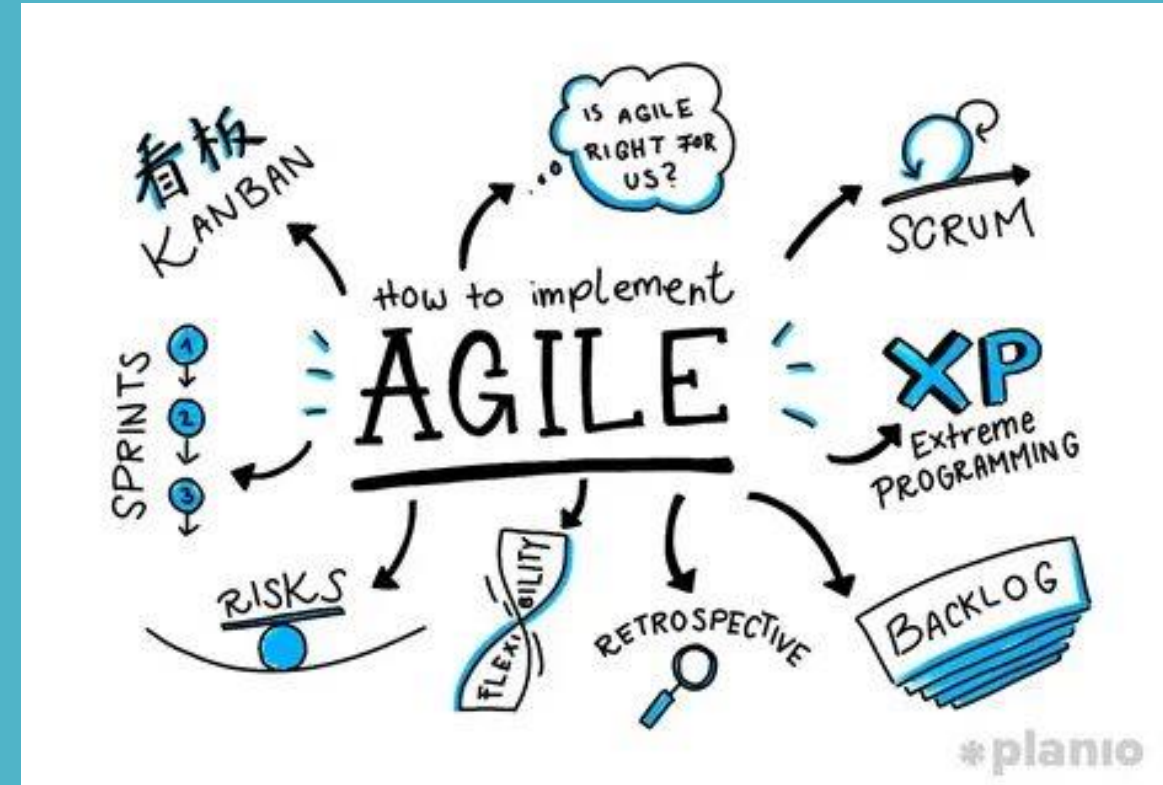


Введени е в Agile.



План занятия:

1. Основные принципы Agile-разработки.
2. Введение в Scrum и Kanban.
3. Инструменты управления задачами и проектами



1. Основные принципы Agile-разработки.

Agile-разработка является гибким и итеративным подходом к разработке программного обеспечения, который акцентирует внимание на быстрой адаптации, коллаборации и постоянном улучшении процесса разработки. Основные принципы Agile-разработки включают:

Сотрудничество с заказчиком (Customer Collaboration) - активное взаимодействие и постоянное сотрудничество с представителями заказчика или конечными пользователями. Это позволяет лучше понять их потребности, получать обратную связь и вовлекать их в процесс разработки.

Готовность к изменениям (Responding to Change) - гибкость и способность быстро адаптироваться к изменяющимся требованиям и условиям. Agile-команды ценят изменения как возможность для улучшения и стремятся к гибкой архитектуре и процессам, чтобы легко вносить изменения в проект.

Итеративность и инкрементальность (Iterative and Incremental) - разработка программного обеспечения осуществляется через короткие циклы разработки, называемые итерациями или спринтами. Каждая итерация добавляет новый функционал и приводит к выпуску инкремента, который может быть протестирован и предоставлен заказчику.

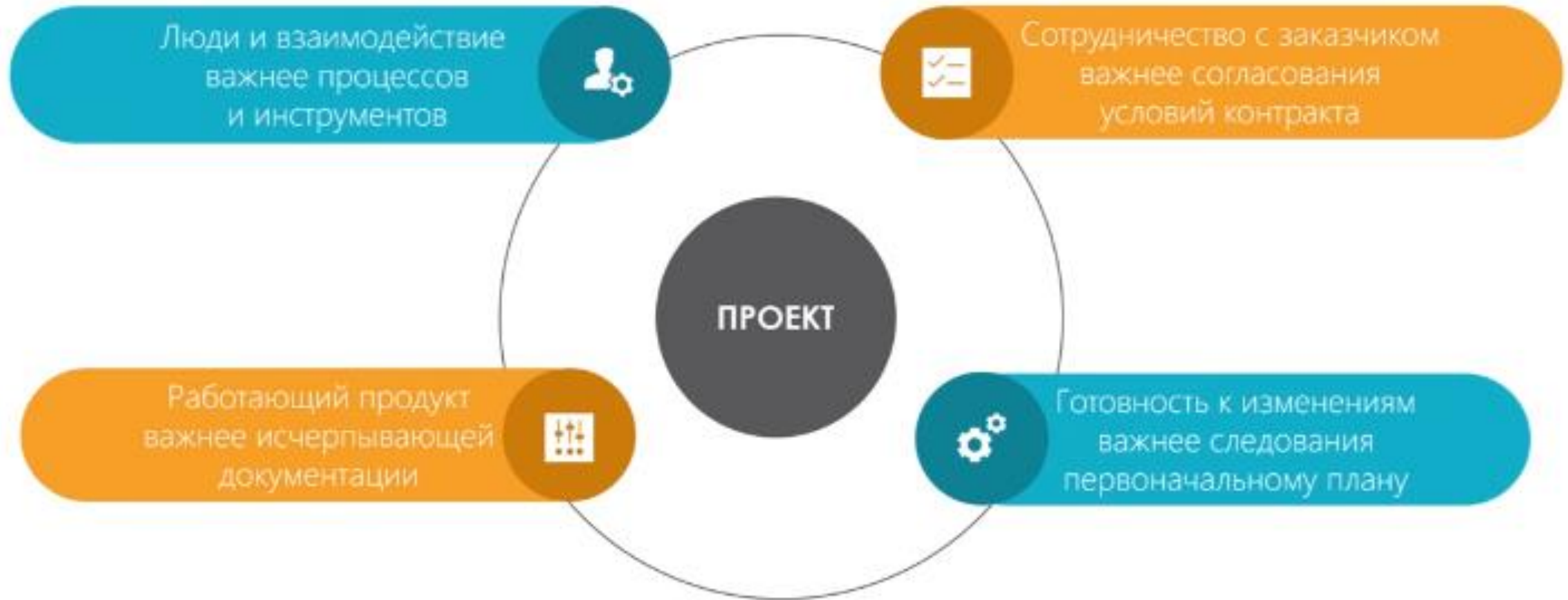
Самоорганизация и коллективная ответственность (Self-organization and Collective Responsibility) - команды Agile имеют высокую степень самоорганизации, где каждый член команды имеет возможность принимать решения и вносить вклад в процесс разработки. Коллективная ответственность означает, что команда вместе несет ответственность за достижение результатов и достижение общих целей.

Постоянное улучшение (Continuous Improvement) - Agile-команды стремятся к непрерывному улучшению процесса разработки. Они регулярно проводят ретроспективы, чтобы анализировать прошлый опыт, выявлять проблемы и находить способы улучшения качества и эффективности работы.

Быстрые результаты (Deliver Working Software Frequently) - Agile-разработка стремится к частым выпускам работающего программного обеспечения. Упор делается на создание ценности для заказчика и обеспечение быстрой обратной связи.

Прозрачность (Transparency) - Agile-команды стремятся к открытости и прозрачности в отношении процессов разработки, прогресса и проблем. Это способствует лучшему пониманию и сотрудничеству в команде и с заказчиком.

AGILE MANIFESTO



2. Введение в Scrum и Kanban

Scrum и Kanban - это две из наиболее популярных методологий Agile-разработки, которые используются для управления проектами и разработки программного обеспечения. Обе методологии призваны повысить эффективность, прозрачность и гибкость в процессе разработки, позволяя командам более адаптивно реагировать на изменения и быстрее доставлять ценное программное обеспечение.

2.1 Scrum



Scrum является одной из самых широко используемых методологий Agile. Его основная идея заключается в организации работы в виде коротких, итеративных периодов, называемых **спринтами**. Каждый **спринт** обычно длится от 1 до 4 недель.



Основные понятия Scrum:

1. Product Backlog:

Product Backlog представляет собой упорядоченный список всех требований, функциональностей, изменений и улучшений, которые должны быть реализованы в продукте.

Он является одним из ключевых артефактов Scrum и создается и поддерживается Product Owner'ом (владельцем продукта).

Product Backlog постоянно эволюционирует и обновляется в процессе разработки продукта, и новые элементы могут быть добавлены или существующие изменены или удалены.

Каждый элемент Product Backlog'a должен быть ясно определен и иметь оценку сложности или размера (например, через user story points), чтобы облегчить планирование и приоритизацию.

2.Sprint Backlog:

Sprint Backlog представляет собой список задач или работ, которые команда разработки выбирает из Product Backlog'a для реализации во время конкретного спринта (итерации).

Sprint Backlog создается командой разработки на основе приоритетов, определенных Product Owner'ом.

Задачи в Sprint Backlog'e должны быть ясными, измеримыми и понятными, чтобы команда разработки могла понять, какие работы должны быть выполнены и как они будут реализованы.

Каждый элемент Sprint Backlog'a обычно имеет оценку трудоемкости или прогнозируемое время выполнения.

3.Increment:

Increment представляет собой накопленную итоговую работу команды разработки в течение спринта.

Он должен быть полностью функциональным и готовым к представлению заказчику или пользователю.

Каждый спринт должен создавать новый Increment, который может быть протестирован и проверен на соответствие требованиям и критериям приемки.

Increment должен быть "готовым" (Done) со всеми необходимыми тестами, документацией и качеством, чтобы быть готовым к релизу или использованию.

Каждый новый Increment строится на предыдущих итерациях, что позволяет продукту постепенно развиваться и расти.

Основные концепции Scrum включают:

Роли

Спринты и планирование

Ежедневные совещания

Демонстрация

Ретроспектива

Scrum. Роли:

- 1. **Scrum Master:** Обеспечивает следование Scrum-процессу, помогает устранять препятствия и обеспечивает эффективную командную работу.
- 2. **Product Owner:** Определяет требования и приоритеты, управляет бэклогом продукта.
- 2. **Команда :** Отвечает за создание и доставку работающего программного продукта.

Scrum Master:

Scrum-мастер – это своеобразный коуч для коллектива. Он поддерживает участников команды, распределяет между ними задачи, дает необходимые инструкции. Его главная цель – внедрить в работу компании scrum-принципы, научить команду самоорганизации и самоконтролю.

Основные обязанности Scrum Mastera :

Обеспечение правильного применения методологии Scrum.

Устранение препятствий, которые мешают команде достичь своих целей.

Помощь в управлении коммуникацией и взаимодействием внутри команды.

Организация и проведение ежедневных стендап-митингов (stand-up meetings), ретроспектив (retrospectives) и планирования спринтов (sprint planning).

Обучение и поддержка команды в применении Scrum-принципов и практик.

Product Owner:

Роль Product Ownera представляет интересы заказчика или бизнеса и отвечает за создание и поддержку видения продукта.

Основные обязанности Product Owner включают:

Определение требований и приоритетов продукта.

Создание и поддержка Product Backloga - списка задач и функциональностей, которые должны быть реализованы.

Обеспечение четкого понимания требований и ожиданий заказчика командой разработчиков.

Принятие решений о содержимом и выпуске инкрементов продукта.

Активное взаимодействие с командой разработчиков и обратная связь по продукту.

Команда разработчиков:

Команда разработчиков является группой специалистов, ответственных за реализацию требований и доставку работающего программного обеспечения.

Основные характеристики команды разработчиков включают:

Самоорганизация и коллективная ответственность за достижение целей спринта.

Разнообразие навыков и экспертизы, которые необходимы для разработки и тестирования продукта.

Работа по итеративному подходу, в рамках спринтов.

Взаимодействие с Product Ownerом для уточнения требований и обратной связи.

Регулярное обсуждение прогресса и проблем на ежедневных стендап-митингах и ретроспективах.

Scrum. Спринты и планирование:

Спринты: Итеративные периоды, в течение которых команда выполняет задачи и создает инкремент продукта.

Планирование спринта: Определение задач, которые команда планирует завершить в следующем спринте.

Планирование спринтов:

Спринт в Scrum-методологии представляет собой фиксированный временной интервал, обычно от 1 до 4 недель, в течение которого команда разработчиков выполняет работу над задачами из Product Backloga. Спринт является основной итерацией в Scrum и обеспечивает фокусировку работы команды и регулярную доставку работающего инкремента продукта.

Этапы планирования:

Подготовка:

Перед началом планирования спринта Product Owner и команда разработчиков собираются для подготовительных действий. Product Owner обновляет Product Backlog, добавляя новые задачи и уточняя приоритеты. Команда разработчиков просматривает Product Backlog и проводит анализ требований.

Определение цели спринта:

На этом этапе Product Owner и команда разработчиков определяют цель или набор целей, которые они планируют достичь в течение спринта. Цель спринта должна быть конкретной, измеримой и достижимой.

Выбор задач:

Команда разработчиков выбирает задачи из Product Backloga, которые они планируют выполнить в течение спринта. Они обсуждают объем работ, оценивают сложность и учитывают свою пропускную способность. Обычно задачи выбираются таким образом, чтобы достичь цели спринта.

Разбиение задач:

После выбора задач команда разработчиков разбивает их на более мелкие и конкретные задания, которые можно легко оценить и выполнить в течение спринта. Это помогает лучше спланировать и распределить работу между членами команды.

Оценка и планирование:

На этом этапе команда разработчиков оценивает каждую задачу по времени и сложности. Оценки могут быть выражены в часах, днях или других единицах измерения времени. Команда планирует, сколько задач они могут выполнить в течение спринта, и создает Sprint Backlog - список задач, которые будут выполнены в спринте.

Завершение планирования:

По завершении планирования спринта команда разработчиков и Product Owner проводят финальное согласование и утверждают Sprint Backlog. Все члены команды должны иметь понимание и согласие с планом спринта.

Планирование спринтов:

После завершения планирования спринта команда разработчиков начинает работу над задачами из Sprint Backlogа.

В течение спринта они проводят ежедневные стендап-митинги, отслеживают прогресс и решают препятствия.

По завершении спринта проводится Sprint Review и ретроспектива, что завершает текущий спринт и подготавливает команду к следующему.

Управление спринтами в Scrum

Роли в управлении спринтами:

Product Owner: Определяет требования и приоритеты, работает с командой разработчиков и принимает решения относительно содержания спринта.

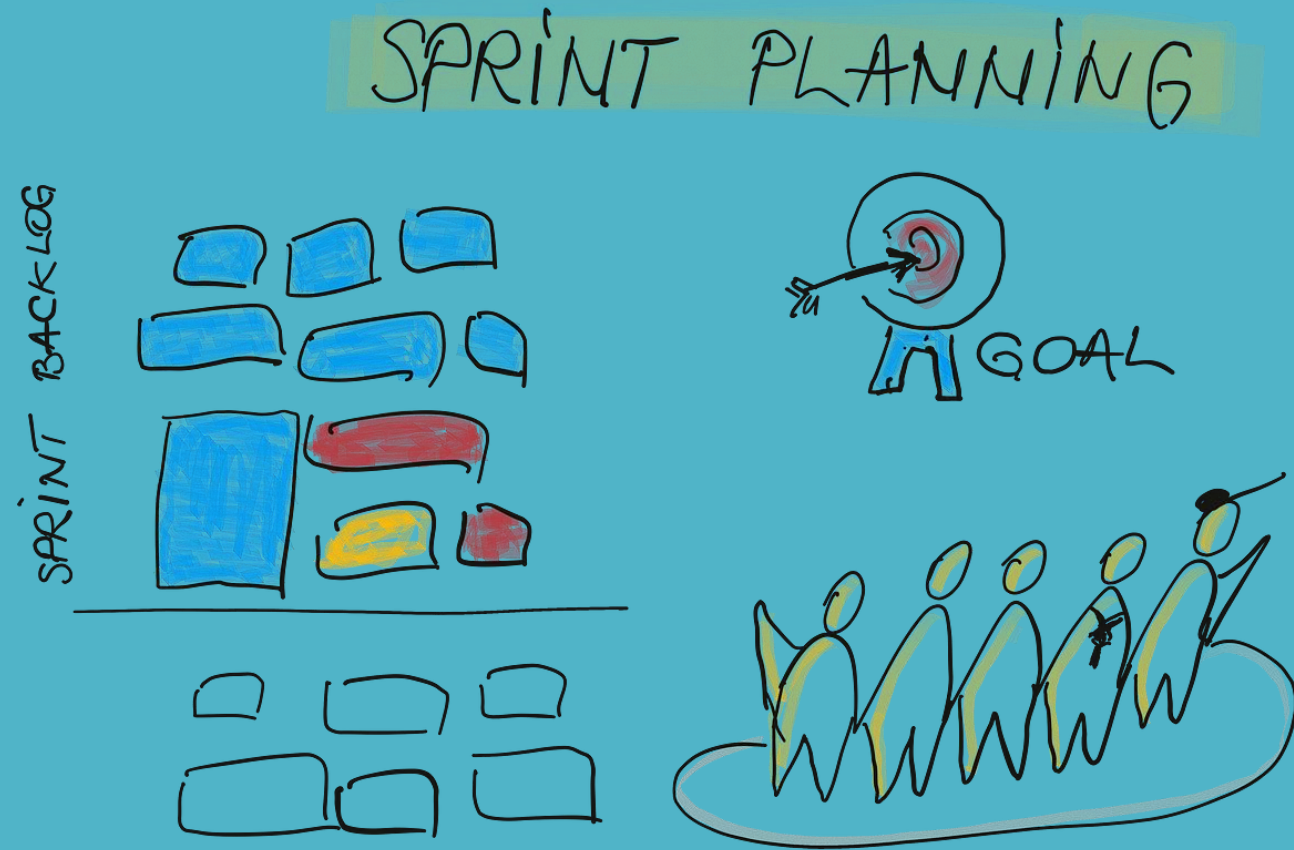
Scrum Master: Обеспечивает соблюдение принципов и практик Scrum, помогает команде разработчиков и Product Owner в достижении целей спринта.

Команда разработчиков: Отвечает за выполнение работ в рамках спринта и доставку готового инкремента продукта.

Управление спринтами в Scrum

Sprint Planning (Планирование спринта):

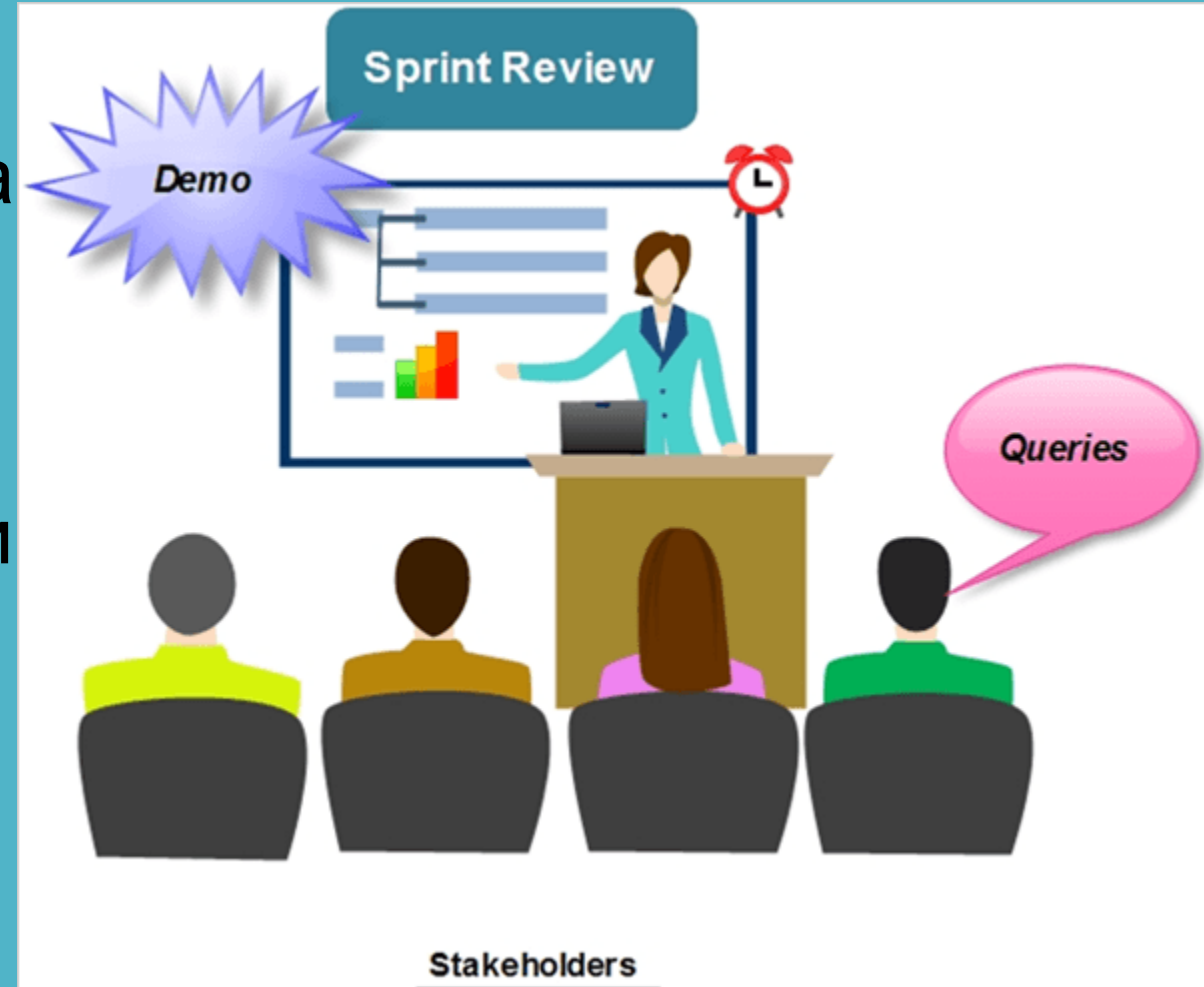
На этом этапе Product Owner и команда разработчиков совместно определяют цели и объем работ на следующий спринт. Они обсуждают требования, выбирают задачи для спринта и создают Sprint Backlog.



Управление спринтами в Scrum

Sprint Review (Обзор спринта):

По завершении спринта команда разработчиков представляет готовый инкремент продукта. На Sprint Review команда и заинтересованные стороны обсуждают результаты спринта и представленный инкремент. При необходимости Product Backlog может быть обновлен на основе обратной связи.



Управление спринтами в Scrum

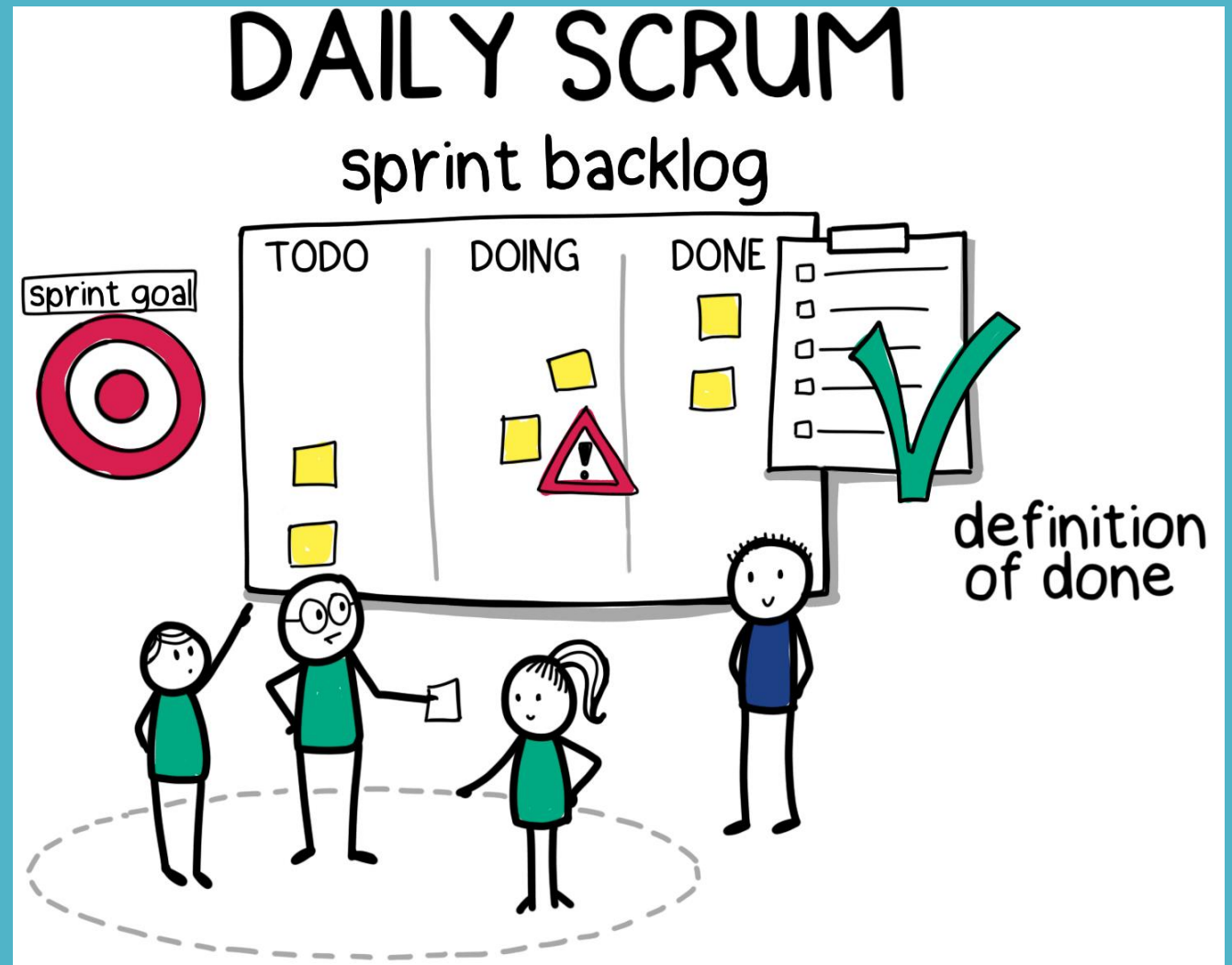
Мониторинг и прогресс:

В течение спринта команда разработчиков и Scrum Master отслеживают прогресс выполнения задач. Они обсуждают препятствия и проблемы, которые могут возникнуть, и предпринимают меры для их решения.

Управление спринтами в Scrum основано на итеративности, прозрачности и коллаборации. Это позволяет команде работать эффективно, доставлять работающий продукт в короткие сроки и непрерывно улучшать процесс разработки.

Scrum. Ежедневные совещания:

Daily Scrum (ежедневный митинг): Короткие совещания команды, на которых каждый участник отчитывается о своем прогрессе, препятствиях и планах на день.



Daily Scrum (Ежедневное совещание):

Ежедневные стендап-митинги проводятся для синхронизации работы команды разработчиков. Каждый член команды докладывает о своем прогрессе, проблемах и планах на ближайшее время. Цель встречи - обеспечить прозрачность и координацию работ.

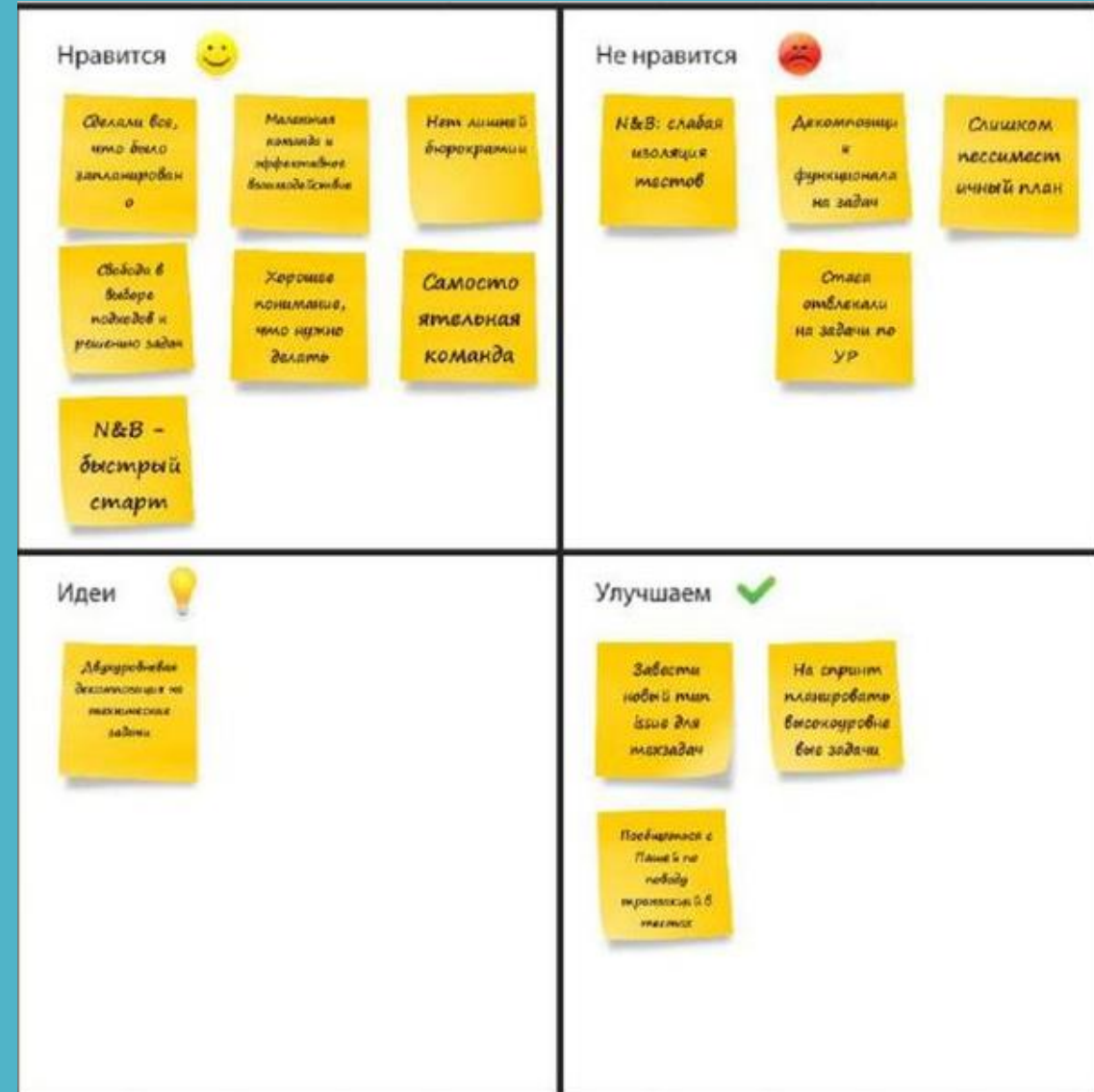
Scrum. Демонстрация

Демонстрация (Sprint Review): Показ работающего релиза продукта по завершению спринта.



Scrum. Ретроспектива:

Ретроспектива: Анализ прошлого спринта с целью улучшения процесса в будущем.



Sprint Retrospective (Петроспектива спринта):

Петроспектива проводится после Sprint Review и направлена на анализ прошлого спринта и поиск способов улучшения процесса. Команда разработчиков обсуждает, что работало хорошо, что можно улучшить и формулирует план действий на следующий спринт.

Scram-доска

БЭЖАОГ	нужно сделать	ДЕЛАЮ	ПРОВЕРКА	ГОТОВО
	 			 
	 		 	
		 		 

Scrum. Заключение.

Scrum позволяет гибко адаптироваться к изменениям. Если в ходе спринта появляются новые требования или приоритеты меняются, Product Owner может внести корректировки в Product Backlog, и эти изменения могут быть учтены в следующих спринтах.

П.С: SCRUM снижает время на разработку, увеличивает время устранения ошибок, убирает лишние процессы и снижает издержки. Хорошо подходит для небольших стартапов, когда важно быстро выпустить релиз.

Пример работы команды по методологии Scrum:

1. Планирование спринта:

1. Команда (разработчики, тестировщики, дизайнеры и т.д.) и владелец продукта (Product Owner) собираются на планировании спринта.
2. Владелец продукта представляет список задач, известный как Product Backlog.
3. Команда и владелец продукта обсуждают задачи и приоритеты, а затем команда выбирает задачи, которые она сможет выполнить в течение одного спринта (обычно 1-2 недели).
4. Выбранные задачи переносятся в Sprint Backlog.

2. Выполнение спринта:

1. Команда ежедневно проводит краткое совещание, называемое Daily Scrum или Stand-up, чтобы обсудить прогресс, проблемы и планы на день.
2. Каждый член команды работает над своими задачами, применяя принципы коллективной работы и самоорганизации.
3. Владелец продукта доступен для команды, чтобы отвечать на вопросы и предоставлять уточнения по требованиям продукта.
4. Команда стремится выполнить все задачи, перенесенные в Sprint Backlog, к концу спринта.

3. Обзор спринта:

1. По окончании спринта команда проводит обзор спринта (Sprint Review) с владельцем продукта и заинтересованными сторонами.
2. Команда представляет завершенную работу и продемонстрирует функциональность, которая была добавлена в приложение.
3. Владелец продукта и заинтересованные стороны задают вопросы, выражают свое мнение и предлагают дополнительные требования или изменения.

4. Ретроспектива спринта:

1. После обзора спринта команда проводит ретроспективу спринта (Sprint Retrospective), чтобы оценить свою работу и идентифицировать улучшения.
2. Команда обсуждает, что работало хорошо, какие проблемы возникли и как можно улучшить свой процесс работы в следующих спринтах.
3. Команда определяет конкретные действия по улучшению и вносит соответствующие изменения в свой рабочий процесс.

2.2 Kanban

Kanban - это методология, которая сосредотачивается на визуализации и оптимизации потока работы. В отличие от Scrum, Kanban не имеет фиксированных спринтов и итераций.



Основные принципы Kanban :

Визуализация потока работы:

Доска Kanban: Визуальное представление задач и их состояний.

Колонки: Столбцы на доске, представляющие различные этапы работы над задачами.

Ограничение Work in Progress (WIP):

WIP-лимит (work in progress) — это число, которое показывает максимальное количество задач в определенной области доски. Эти лимиты могут устанавливаться на человека, на этап работ или на всю команду. Например, если у команды WIP-лимит равен четырем, то у нее в работе может находиться не больше четырех задач одновременно:

Если лимит закончился, команда должна посмотреть на доску и попробовать продвинуть вправо какую-нибудь из текущих задач.

Постоянное улучшение:

Усовершенствование потока работы на основе наблюдения за процессом и регулярного анализа.

Непрерывная доставка:

Более гибкий и непрерывный процесс доставки, поскольку задачи могут двигаться по доске в любое время.

Управление потоком:

Основное внимание уделяется поддержанию стабильного потока работы и сокращению времени между этапами.

Доска Канбан



Доска Kanban:

- Доска Kanban представляет собой физическую или виртуальную доску, на которой отображаются задачи или работа, которые нужно выполнить, в процессе выполнения и уже завершённые задачи.
- Обычно доска разделена на несколько колонок, представляющих различные этапы рабочего процесса. Наиболее распространёнными колонками являются "Запланировано", "В процессе", "Готово" или "В ожидании проверки", но конкретные колонки могут быть адаптированы под потребности конкретной команды или проекта.
- Каждая задача представлена карточкой, которая содержит информацию о задаче, её статусе, ответственном лице и других деталях, которые могут быть полезными для команды.
- Карточки перемещаются по колонкам доски, отражая текущий статус работы над задачами. Например, карточка может начинаться в колонке "Запланировано", затем перемещаться в колонку "В процессе" и, наконец, в колонку "Готово", когда задача завершена.

Визуализация рабочего процесса:

Визуализация рабочего процесса в методологии Kanban позволяет команде видеть и отслеживать весь поток работы в проекте.

Каждая колонка на доске представляет этап рабочего процесса, и перемещение карточек отражает передачу работы от одного этапа к другому.

Визуализация рабочего процесса помогает команде лучше понять, где задачи застревают, где возникают задержки, и как можно оптимизировать рабочий поток.

Это также способствует прозрачности и общему пониманию работы в команде. Все члены команды могут видеть, какие задачи активны, какие завершены и какие задачи находятся в очереди.

KANBAN. Пример работы команды над проектом

состав команды:

Владелец продукта (Product Owner): Ответственный за определение требований, управление бэклогом и обеспечение ценности продукта.

Разработчики (Developers): Отвечают за разработку мобильного приложения, написание кода, тестирование и интеграцию.

Дизайнеры (Designers): Занимаются созданием пользовательского интерфейса, визуальным оформлением и пользовательским опытом.

Тестировщики (Testers): Отвечают за тестирование приложения и обеспечение его качества.

Аналитики (Analysts): Занимаются анализом требований, сбором и анализом данных, а также оценкой производительности приложения.

DevOps-инженеры: Отвечают за автоматизацию процессов разработки, развертывание и обеспечение непрерывной поставки приложения.

Канбан

Фильтры

тема или ссылка



НОВАЯ



ГОТОВО



В ПРОЦЕССЕ



МОЖНО ПРОВЕРЯТЬ



ЗАВЕРШЕНА



#5 Реализовать систему комментариев к задачам



Не назначен

#6 Разработать систему отправки уведомлений на почту



Не назначен

#3 Добавить возможность прикрепления файлов к задаче



Не назначен

#4 Уведомления и Оповещения



Степан

#7 Создание и Редактирование Задач



Руслан

#1 Создать дизайн главной страницы



Арина

пояснения к каждому из столбцов на доске Kanban:

1.Backlog (Задачи в очереди): В этом столбце размещаются все задачи, которые планируется реализовать в будущем. Это своего рода "хранилище" идей и задач, которые ожидают своей очереди на выполнение. Здесь задачи еще не разбиты на более детальные подзадачи и не назначены конкретным исполнителям. Этот столбец позволяет команде видеть полный перечень задач, из которого можно выбирать следующую для выполнения.

2.To Do (Задачи в работе): Когда команда приняла решение начать работу над определенной задачей из "Backlog", она перемещается в столбец "To Do". Здесь каждая задача уже разбивается на более мелкие подзадачи, назначаются исполнители и присваиваются сроки выполнения. Эти подзадачи готовы к реализации.

3.In Progress (Задачи в процессе): В этом столбце размещаются задачи, над которыми в данный момент активно работают члены команды. Они уже начали реализацию подзадач и работают над их выполнением. Задачи в этом столбце находятся в процессе выполнения и обычно имеют назначенных исполнителей.

4.In Progress (Задачи в процессе): В этом столбце размещаются задачи, над которыми в данный момент активно работают члены команды. Эти задачи проходят этап активной реализации и разработки. Они могут включать в себя различные подзадачи и этапы работы.

5.Testing (Тестирование): Когда разработчики завершают работу над задачей и считают ее готовой, она перемещается в столбец "Testing". Здесь задача подвергается тестированию для обеспечения качества. Это может включать в себя функциональное, интеграционное, и другие виды тестирования.

Пример действий:

1. В начале проекта, команда собралась и заполнила "Backlog" всеми задачами, которые требуется реализовать.
2. Каждый участник команды выбрал задачу из "To Do" и начал работу над ней.
3. Когда задача начата, она переносится в "In Progress" и помечается именем того, кто работает над ней.
4. После завершения работы над задачей, она переносится в "Testing" для проверки качества.
5. Если задача прошла тестирование, она перемещается в "Done".

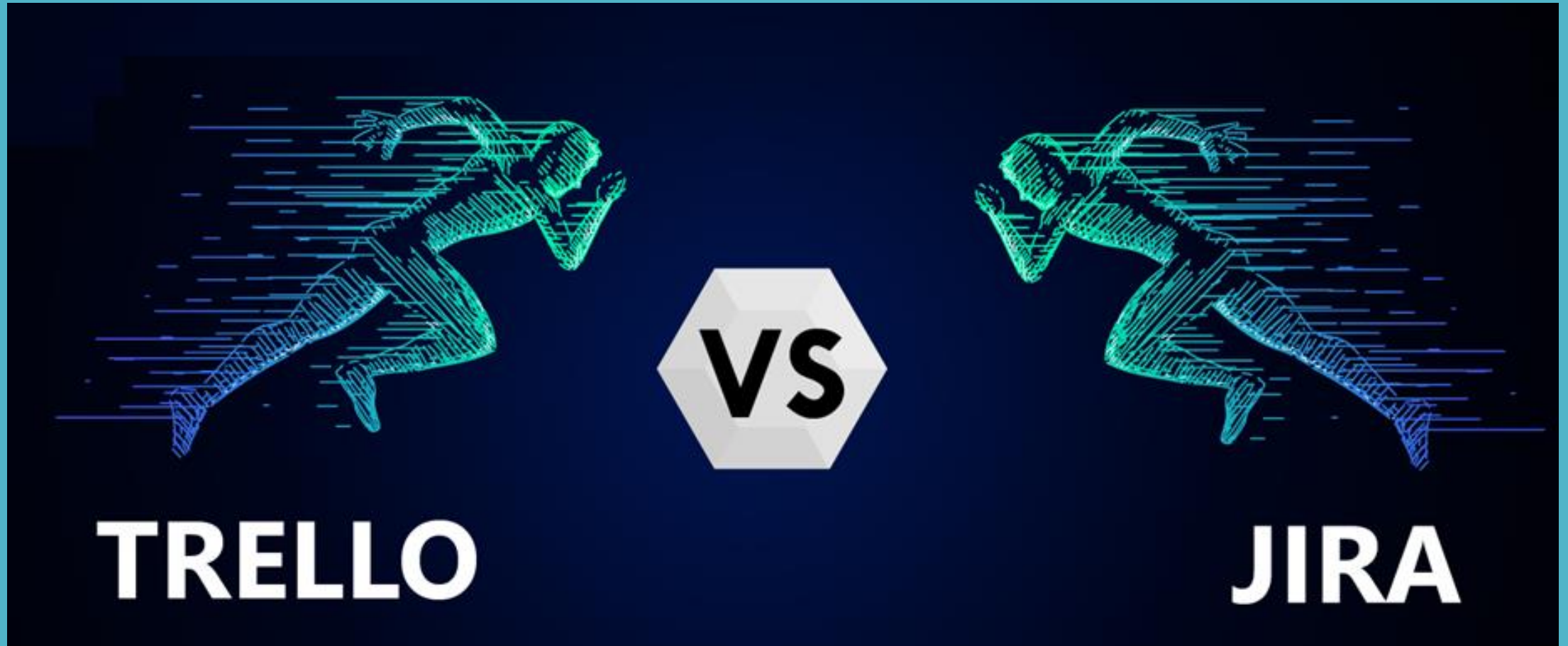
Примечания:

- Каждый день команда проводит короткие встречи (Daily Standup), на которых обсуждаются прогресс и проблемы.
- В случае возникновения новой важной задачи, она может быть добавлена в "Backlog" и приоритетно взята в работу.

Выбор между Scrum и Kanban может зависеть от потребностей команды и характера проекта. Scrum часто более подходит для команд, которым нравится стройное расписание спринтов, а Kanban хорошо подходит для команд, работающих с постоянным потоком задач.

Выбор между Scrum и Kanban может зависеть от потребностей команды и характера проекта. Scrum часто более подходит для команд, которым нравится стройное расписание спринтов, а Kanban хорошо подходит для команд, работающих с постоянным потоком задач.

3. Инструменты управления задачами и проектами



Эффективное управление задачами и проектами в программной разработке является фундаментальным элементом для достижения успеха в проектах. Оно способствует организации работы, оптимизации ресурсов, соблюдению сроков, повышению качества и прозрачности процесса, а также обеспечивает гибкость и адаптацию к изменениям внешних условий.

Инструменты управления проектами - это программное решение или набор программных систем, предназначенных для помощи командам, участвующим в проекте. Они способствуют облегчению работы менеджеров проекта, ее стандартизации, а также повышению эффективности в вопросах организации рабочего процесса и управления задачами проекта.

Инструменты управления задачами — это программные приложения, платформы или методологии, которые помогают упорядочить, организовать и отслеживать задачи, которые нужно выполнить в рамках проекта или работы. Они предоставляют средства для создания, назначения, отслеживания и управления задачами, а также для контроля прогресса и координации работы между участниками команды.

А. Традиционные инструменты

1. Таблицы и электронные таблицы (например, Microsoft Excel, Google Sheets).
2. Заметки и списки дел (например, Microsoft Outlook, Google Keep).








В. Канбан-доски

Канбан-доска - это визуальный инструмент управления проектами, который помогает следить за текущим положением дел организации и упрощает общение в команде.

Канбан-доска представляет собой доску проекта, которая делится на столбцы.

Обычно каждый столбец представляет собой некий этап работы.

Отдельные задачи, которые представлены на доске в виде карточек, перемещаются из одного столбца в другой по мере их выполнения

ОЧЕРЕДЬ		АНАЛИЗ	РАЗРАБОТКА	ТЕСТИРОВАНИЕ	ПРИЕМ. ТЕСТ.	ГОТОВО!!
	ВАДИМ					
	АНДРЕЙ					
	ДИМА					
	СЛАВУК					

Некоторые популярные канбан-инструменты



Некоторые популярные канбан-инструменты

	Тип версии	Ключевые ограничения	Пользователи	Проекты/доски	Задачи	Место на диске
YouGile	честная	нет	до 10	сколько угодно	сколько угодно	сколько угодно
Trello	честная	права доступа, сортировка карточек	сколько угодно	до 10 досок	сколько угодно	10 МБ
Bitrix24	честная	интеграции, бизнес-процессы	сколько угодно	сколько угодно	сколько угодно	5 ГБ
Pyrus	честная	до 100 задач по формам	сколько угодно	сколько угодно	до 100 (по формам)	1 ГБ
Jira	честная	права доступа, журнал событий	до 10	сколько угодно	сколько угодно	2 ГБ
ClickUp	нечестная	лимит на 100 действий	сколько угодно	сколько угодно	сколько угодно	100 МБ
Wrike	нечестная	права доступа, отчеты, контроль	до 5	сколько угодно	сколько угодно	2 ГБ
Asana	нечестная	отчеты, контроль, обновления	до 15	сколько угодно	сколько угодно	100 МБ
Worksection	нечестная	проекты, отчеты, права доступа	до 5	до 2	сколько угодно	100 МБ
ToDoist	нечестная	комментарии, обмен файлами	до 5	до 80	до 150	0

С. Инструменты управления задачами и проектами для разработчиков

Интегрированная среда разработки (IDE) с функциональностью управления задачами представляет собой программное приложение, которое объединяет средства разработки кода с инструментами для организации и отслеживания задач в процессе разработки программного обеспечения. Такие IDE обеспечивают комфортное и эффективное рабочее окружение для разработчиков, интегрируя в себя различные инструменты управления задачами.

IDE с функциональностью управления задачами

JetBrains IntelliJ IDEA: является одной из наиболее популярных IDE для разработки на языках Java, Kotlin, Scala и других. Она предлагает встроенные инструменты для управления задачами, такие как системы отслеживания ошибок, интеграция с системами контроля версий, плагины для поддержки различных методологий управления задачами и возможность создания и просмотра списка задач.

JetBrains PyCharm предлагает встроенные инструменты для управления задачами, которые помогают разработчикам организовывать и отслеживать свою работу. В IDE есть возможность создавать задачи, устанавливать им статусы, приоритеты и сроки выполнения. Вы также можете назначать задачи себе или другим участникам команды и отслеживать их прогресс. PyCharm интегрируется с системами отслеживания ошибок и системами контроля версий. Поддерживает интеграцию с популярными инструментами управления задачами, такими как JIRA, Trello, Asana и другими.

Microsoft Visual Studio: является мощной IDE для разработки на платформе Microsoft. Она предлагает функциональность управления задачами через интегрированные инструменты, такие как Azure DevOps или Team Foundation Server. Разработчики могут создавать, назначать и отслеживать задачи, а также видеть их статус и сроки выполнения прямо в IDE.

Eclipse: распространенная IDE, особенно популярная в сообществе разработчиков Java. Она также предлагает плагины и расширения для управления задачами, такие как Mylyn. Mylyn позволяет создавать и отслеживать задачи, связывать их с определенными файлами или кодовыми фрагментами, а также интегрироваться с системами отслеживания ошибок и системами контроля версий.

Xcode: IDE, разработанная компанией Apple для создания приложений под iOS, macOS и другие платформы Apple. Xcode включает инструменты, позволяющие разработчикам создавать и управлять задачами, а также интегрироваться с системами отслеживания ошибок, такими как Bugzilla или JIRA.

Инструменты управления версиями

Инструменты управления версиями (Version Control Tools) представляют собой программные средства, которые помогают разработчикам отслеживать и управлять изменениями в исходном коде и других файловых ресурсах во время разработки программного обеспечения. Они позволяют командам разработчиков сотрудничать, отслеживать изменения, возвращаться к предыдущим версиям и управлять конфликтами при одновременном редактировании файлов.

Git - одна из самых популярных распределенных систем управления версиями. Он отслеживает изменения в репозитории, позволяет создавать ветки для параллельной разработки, объединять изменения из разных веток и восстанавливать предыдущие версии. Git также обладает широкой поддержкой и широким набором инструментов и сервисов, таких как GitHub, GitLab и Bitbucket.

Subversion (SVN) - централизованная система управления версиями, где репозиторий находится на центральном сервере.

Mercurial - распределенная системой управления версиями, которая подобна Git.

Perforce - централизованная система управления версиями, которая широко используется в разработке больших проектов.

Инструменты для отслеживания багов и запросов на изменение

Инструменты для отслеживания багов и запросов на изменение (Issue Tracking and Change Request Tools) - программные средства, которые помогают командам разработчиков эффективно управлять и отслеживать проблемы, ошибки (баги) и запросы на изменение в процессе разработки программного обеспечения.

JIRA - один из самых популярных инструментов для управления задачами и отслеживания ошибок. Он предоставляет возможности создания, назначения, отслеживания и приоритизации задач, багов и запросов на изменение.

Bugzilla - бесплатная и открытая система отслеживания ошибок. Она предоставляет возможности создания и отслеживания багов, запросов на изменение и других задач. Bugzilla также обладает расширенными возможностями для управления жизненным циклом ошибки, отчетности и интеграции с другими инструментами разработки.

GitHub Issues - встроенный инструмент отслеживания задач, багов и запросов на изменение в системе контроля версий Git. Он интегрирован с платформой GitHub и предоставляет возможности для создания, назначения, отслеживания и обсуждения задач.

GitLab Issues - встроенный инструмент отслеживания задач в системе контроля версий GitLab. Он предлагает возможности для создания, назначения, отслеживания и обсуждения задач, багов и запросов на изменение.

Trello - это гибкая и простая в использовании онлайн-доска для управления проектами и задачами. В Trello можно создавать карточки для багов, запросов на изменение и других задач, перемещать их по колонкам для отслеживания статуса и назначать ответственных.

Инструменты управления проектами

А. Графическое представление проектов

В. Системы управления проектами

С. Виртуальные командные доски

А. Графическое представление проектов

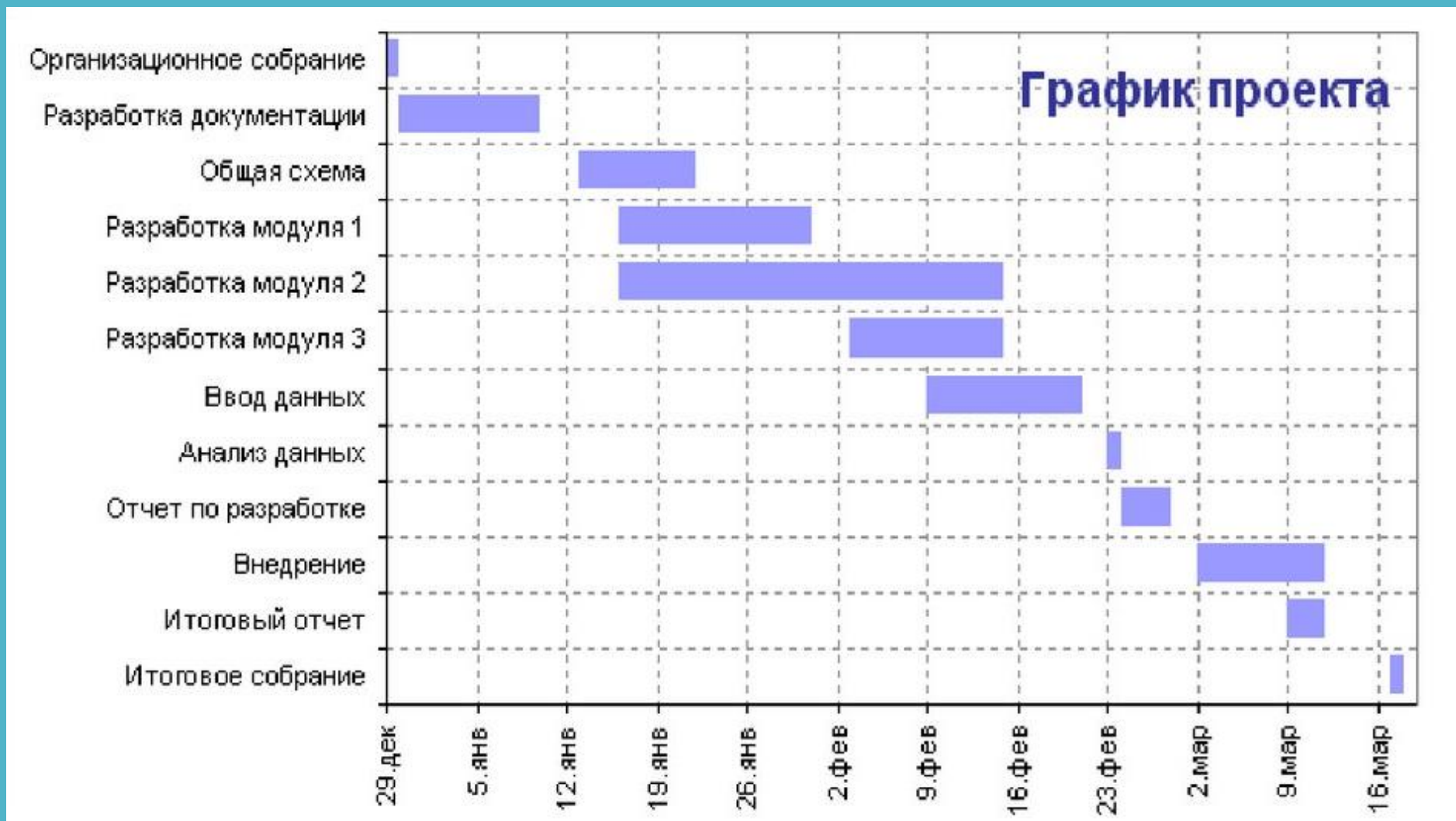


Диаграмма Гранта



Диаграмма PERT
(Program Evaluation and Review Technique)

Программные продукты для управления проектами

Существует множество программных продуктов, специально разработанных для управления проектами. Эти инструменты помогают организовать и координировать задачи, ресурсы и коммуникацию в рамках проекта.



Microsoft Project



Trello



JIRA



asana

Agile-ориентированные инструменты управления проектами

Agile-ориентированные инструменты управления проектами разработаны для поддержки гибких методологий разработки, таких как Scrum, Kanban и др.



Выбор и использование инструментов

Выбор и использование инструментов зависит от нескольких факторов:

1. Целей и потребностей проекта: Важно определить, какие конкретные задачи и цели должны быть достигнуты с помощью инструмента. Некоторые инструменты могут быть лучше всего подходят для определенных типов проектов или методологий разработки.
2. Размера и структуры команды: Размер и структура команды могут влиять на выбор инструмента. Некоторые инструменты могут быть лучше адаптированы для работы небольших команд, в то время как другие могут быть предназначены для масштабирования на большие команды или организации.
3. Бюджета: Стоимость инструмента также может быть фактором выбора. Некоторые инструменты являются платными, в то время как другие предлагают бесплатные или бесплатные версии с ограниченным функционалом.
4. Интеграции и совместимости: Если команда уже использует определенные инструменты для управления проектом, разработки или коммуникации, важно убедиться, что выбранный инструмент может интегрироваться с существующими инструментами или иметь совместимость с ними.
5. Предпочтений команды: Команде должно быть комфортно с выбранным инструментом и находить его удобным для своей работы.

Список литературы:

1. "Управление проектами. Краткий курс" (автор: Рита Мулкахи) - эта книга предлагает введение в основные принципы управления проектами и рассматривает популярные инструменты и методологии, такие как Gantt-диаграммы и методология Agile.
2. "Agile. Гибкое управление проектами" (автор: Андрей Макеев) - в этой книге автор подробно объясняет принципы Agile-разработки и предлагает ряд инструментов и методик, включая Scrum и Kanban.
3. "Управление проектами с использованием Microsoft Project" (автор: Александр Бахмутский) - данная книга ориентирована на использование инструмента Microsoft Project для управления проектами. Она предлагает практические руководства по созданию планов проекта, назначению ресурсов, отслеживанию прогресса и многому другому.
4. "Управление проектами. От идеи до результата" (автор: Елена Кузнецова) - эта книга представляет комплексный подход к управлению проектами, включающий инструменты и методы планирования, контроля, коммуникации и управления рисками.
5. "Scrum. Революционный метод управления проектами" (автор: Джефф Сазерленд) - книга изначально написана на английском языке, она имеет русский перевод и представляет собой важный источник информации о методологии Scrum и его применении в управлении проектами.

ISSUE (Задачи):

Epic – относительно большой объем работ, который состоит из нескольких задач.

Story – задача, выраженная на языке пользователя. часть большой задачи (эпика), которую команда может решить за 1 спринт.

Task – техническая задача, которую делает один из членов команды

Bug - задача, которая описывает ошибку в системе

