Лекция 2

SDLC

Agenda

- Разбор Д/З по требованиям
- Authentication & Authorization
- Жизненный цикл разработки программного обеспечения (SDLC) и место тестирования в нем.
- Модели и методологии разработки ПО
- Домашнее задание

Домашнее задание

Some examples of requirements. How do you think these requirements are good or bad and why?

1. I want to download the list of all users in CSV or Excel.
2. I want to build a support system with live chat, contact form, help and case management
3. I want to store user’s Facebook ID in the database
4. There has to be a way to authenticate
5. The system must have good usability
6. Response time should be less than 5 seconds
7. The system shall work just like the previous one, but on a new platform
8. The system has to be bug-free
9. Easy to use

Authentication & Authorization

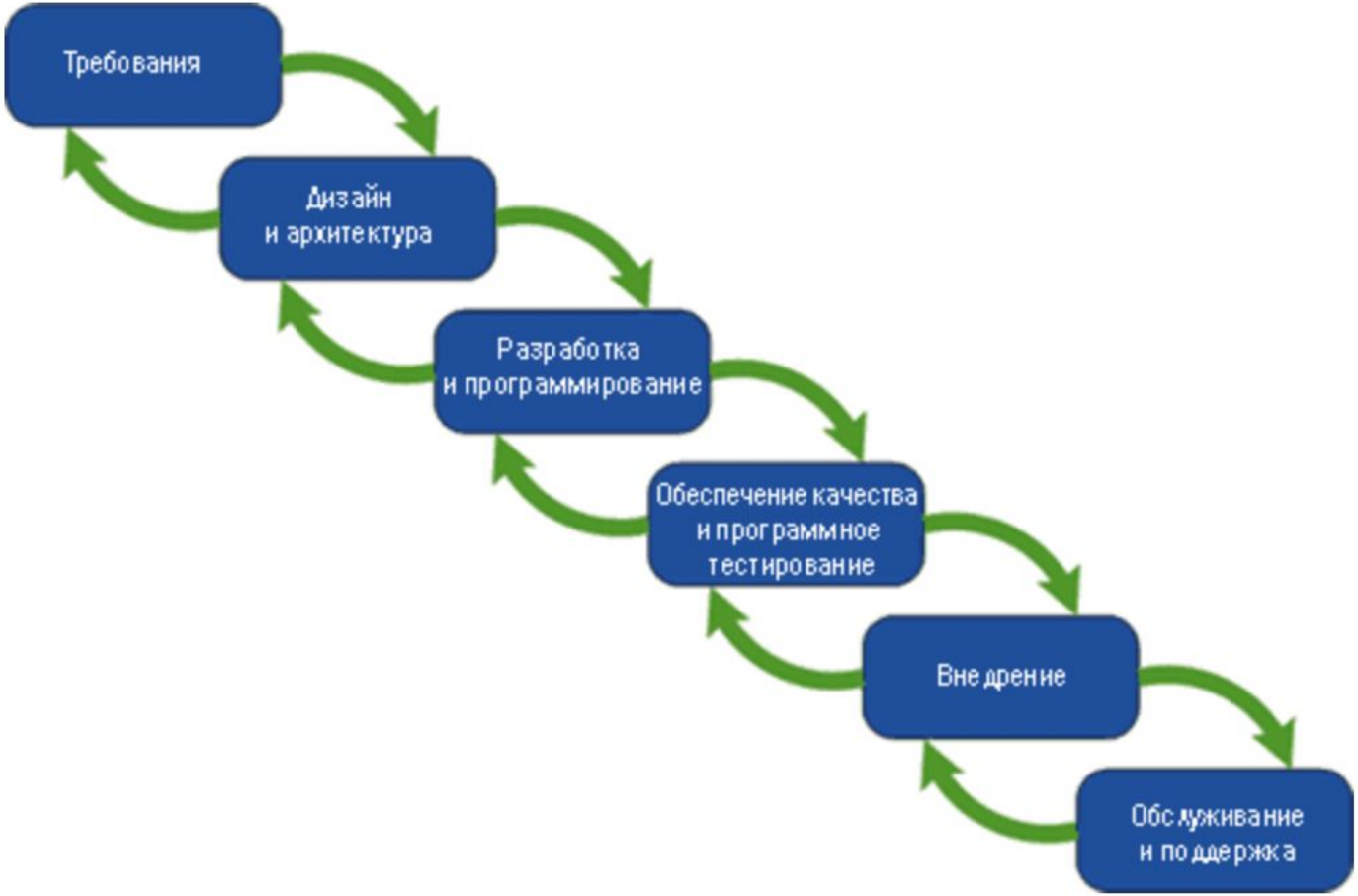
- При аутентификации пользователь или компьютер должны подтвердить свою личность серверу или клиенту. Например, в бухгалтерском отделе все сотрудники могут логиниться и пользоваться 1С, а Катя из юротдела не может.
- Обычно аутентификация на сервере подразумевает использование имени пользователя и пароля. Другими способами аутентификации могут быть карты, сканирование сетчатки глаза, распознавание голоса и отпечатки пальцев.
- Аутентификация не определяет, какие задачи может выполнять человек или какие файлы он может просматривать. Аутентификация просто идентифицирует и проверяет, кем является человек или система.
- Авторизация — это процесс, посредством которого сервер определяет, есть ли у клиента разрешение на использование ресурса или доступ к файлу(ресурсам и тд), то есть определяет уровень доступа.
- Аутентификация проверяет пользователя (Елена-гл.бухгалтер) перед тем, как предоставить ему доступ, а авторизация определяет, что он может делать после того, как система предоставила ему доступ (может редактировать информацию в годовом отчете).
- Аутентификация обеспечивает авторизацию проверенной личности, необходимую для управления доступом.

Жизненный цикл разработки ПО (SDLC)

SDLC – это жизненный цикл разработки программного обеспечения (Software development lifecycle). Он представляет собой несколько этапов (или фаз), которые проходит любое ПО. По сути, это подробный план, показывающий, как разрабатывать программное обеспечение, поддерживать его, изменять, улучшать.

Цикл разработки предлагает шаблон, использование которого облегчает проектирование, создание и выпуск качественного программного обеспечения. Это методология, определяющая процессы и средства, необходимые для успешного завершения проекта.

Цель использования модели жизненного цикла – создать эффективный, экономически выгодный и качественный программный продукт.


Требования
Дизайн
и архитектура
Разработка
и программирование
Обеспечение качества
и программное
тестирование
Внедрение
Обслуживание
и поддержка

Анализ требований

На этом этапе у клиента собирается вся необходимая информация, чтобы разработать продукт в соответствии с его ожиданиями. Любые неясности должны быть разрешены только на этой фазе.

Бизнес-аналитик и менеджер проекта назначают встречу с заказчиком, чтобы собрать всю информацию о том, что заказчик хочет создать, кто будет конечным пользователем, какова цель продукта. Перед созданием продукта очень важно понимание или знание продукта.

После сбора требований проводится анализ, чтобы проверить целесообразность разработки продукта. В случае возникновения неясностей назначается переговоры для дальнейшего обсуждения.

После того как требования четко поняты, создается документ SRS (Software Requirement Specification) или иной. Этот документ должен быть тщательно проработан разработчиками, а также должен быть просмотрен заказчиком для дальнейшего использования.

Цель этой стадии – определение детальных требований к системе. Кроме этого, необходимо убедиться в том, что все участники правильно поняли поставленные задачи и то, как именно каждое требование будет реализовано на практике.

Дизайн и архитектура (проектирование)

На этапе проектирования моделируется то, как будет работать программное приложение. Некоторые аспекты проектирования включают:

Пользовательский интерфейс - определяет способы взаимодействия клиентов с программным обеспечением и то, как программное обеспечение реагирует на вводимые данные.

Платформы - Определяет платформы, на которых будет работать программное обеспечение, например, Apple, Android, Windows, Linux или даже игровые консоли.

Программирование - Не только язык программирования, но и методы решения проблем и выполнения задач в приложении.

Связь - Определяет методы, с помощью которых приложение может взаимодействовать с другими ресурсами, такими как центральный сервер или другие экземпляры приложения.

Безопасность - определяет меры, принятые для обеспечения безопасности приложения, и может включать шифрование трафика, защиту паролем и безопасное хранение учетных данных пользователя.

Прототипирование может быть частью фазы проектирования. Прототип - это как одна из ранних версий программного обеспечения в итерационной модели разработки программного обеспечения. Он демонстрирует основное представление о том, как выглядит и работает приложение. Этот "практический" дизайн можно показать заинтересованным сторонам. Используйте обратную связь для улучшения приложения. Изменения на этапе прототипа обходятся дешевле, чем переписывание кода для внесения изменений на этапе разработки.

Разработка и программирование

Это фактическое написание программы. Небольшой проект может быть написан одним разработчиком, в то время как крупный проект может быть разбит на части и над ним могут работать несколько команд. Процесс кодирования включает в себя множество других задач. Многим разработчикам необходимо подтянуть свои навыки или поработать в команде. Очень важно находить и исправлять ошибки и сбои.

Документирование может быть формальным процессом, включая составление руководства пользователя для приложения. Оно также может быть неформальной, например, комментарии в исходном коде, которые объясняют, почему разработчик использовал ту или иную процедуру. Даже те компании, которые стремятся создавать простое и интуитивно понятное программное обеспечение, получают пользу от документации.

Тестирование

Тестирование начинается после завершения кодирования и выпуска модулей для тестирования. На этом этапе разработанное программное обеспечение тщательно тестируется, и все обнаруженные дефекты передаются разработчикам для их устранения.

Повторное тестирование, регрессионное тестирование проводятся до тех пор, пока программное обеспечение не будет соответствовать ожиданиям заказчика. Тестировщики обращаются к требованиям, чтобы убедиться, что программное обеспечение соответствует стандартам заказчика.

Этап тестирования помогает уменьшить количество ошибок и сбоев, с которыми сталкиваются пользователи. Это приводит к повышению удовлетворенности пользователей и улучшению коэффициента использования.

Внедрение

После того как продукт протестирован, он внедряется в производственную среду или сначала проводится UAT (User Acceptance testing) в зависимости от ожиданий заказчика.

В случае UAT создается копия производственной среды, и заказчик вместе проводит тестирование. Если клиент считает, что приложение соответствует ожиданиям, то заказчик дает согласие на запуск.

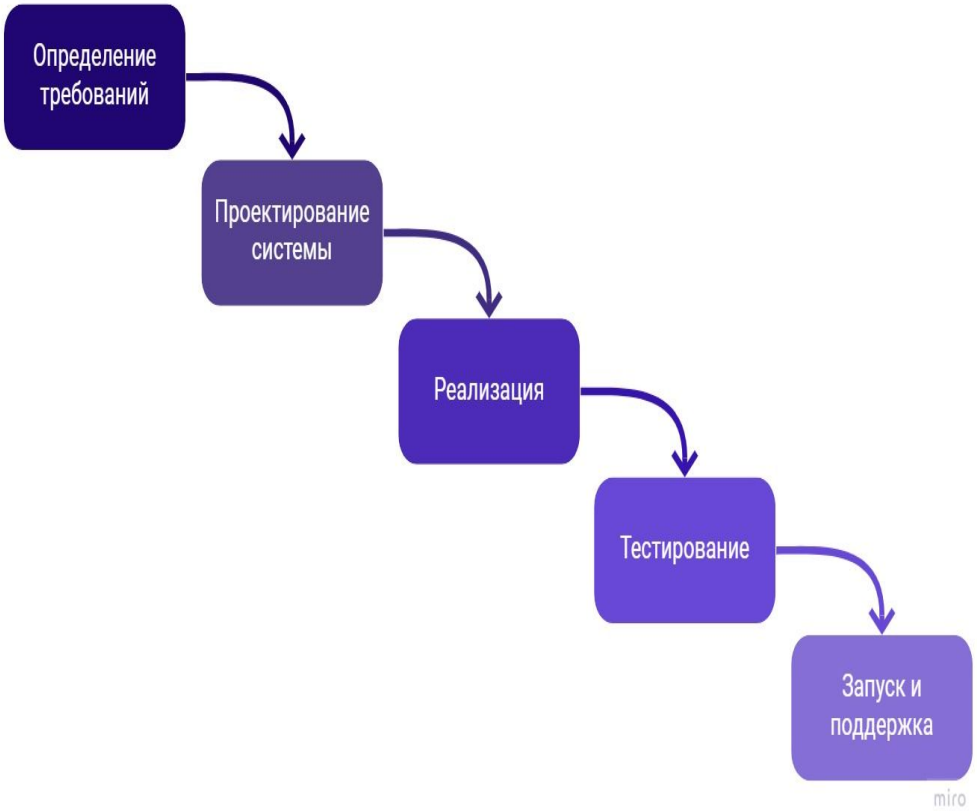
Когда программа протестирована и в ней больше не осталось серьезных дефектов, приходит время релиза и передачи ее конечным пользователям.

Сопровождение

На данном этапе цикл разработки практически завершен. Приложение готово и используется в полевых условиях. Однако фаза эксплуатации и обслуживания все еще важна. На этом этапе пользователи обнаруживают ошибки, которые не были найдены во время тестирования. Эти ошибки необходимо устранить, что может привести к новым циклам разработки.

Модели и методологии разработки ПО

Waterfall (водопадная модель)

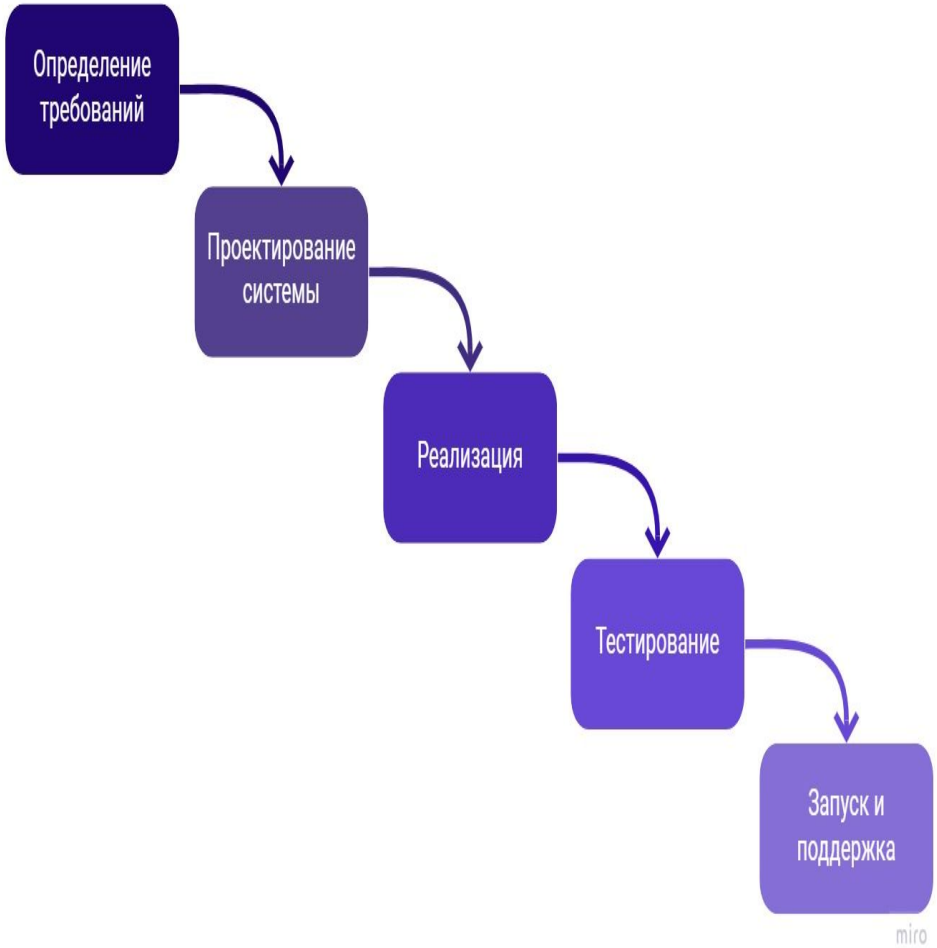


Модель водопада - это самая первая модель, которая используется в SDLC.

В этой модели результат одной фазы является исходным материалом для следующей фазы. Разработка следующей фазы начинается только после завершения предыдущей фазы.

Сначала проводится сбор и анализ требований. Как только требования будут заморожены, можно приступать к проектированию системы.

При проектировании системы создаются документы, которые служат входом для следующей фазы, т.е. реализации и кодирования.



На этапе реализации выполняется кодирование, и разработанное программное обеспечение является исходным материалом для следующего этапа, т.е. тестирования.

На этапе тестирования разработанный код тщательно тестируется для выявления дефектов в программном обеспечении. Дефекты регистрируются, а после их устранения проводится повторное тестирование. Регистрация ошибок, повторное тестирование, регрессионное тестирование продолжаются до тех пор, пока программное обеспечение не будет введено в эксплуатацию.

На этапе развертывания разработанный код передается в производство после получения одобрения заказчика.

Любые проблемы в продакшене решаются разработчиками, что относится к техническому обслуживанию и поддержке.

Преимущества водопадной модели:

Водопадная модель - это простая и понятная модель, в которой все этапы выполняются шаг за шагом

Результаты каждой фазы четко определены, что не приводит к сложностям и делает проект легко управляемым.

Недостатки водопадной модели:

Водопадная модель требует много времени и не может быть использована в проектах небольшой продолжительности, так как в этой модели нельзя начинать новую фазу до завершения текущей фазы.

Водопадная модель не может быть использована для проектов с неопределенными требованиями или требованиями, которые постоянно меняются, так как эта модель предполагает, что требования должны быть четко определены на этапе сбора и анализа требований, а любые изменения на последующих этапах приведут к увеличению затрат, так как изменения потребуются на всех этапах.

Сложность в оценке времени и стоимости процесса, что приводит к задержке поставки продукта.

Agile методологии (Scrum, Kanban,XP)

Модель Agile - это комбинация итеративной и инкрементальной моделей. В этой модели больше внимания уделяется гибкости при разработке продукта, а не требованиям.

В Agile продукт разбивается на небольшие инкрементальные сборки. Он не разрабатывается как полный продукт за один раз. Каждая сборка увеличивается в плане функциональности. Следующая сборка строится на основе предыдущей функциональности.

В конце каждой итерации владелец продукта проверяет продукт, и после его одобрения он передается заказчику.

Обратная связь с клиентом учитывается для улучшения продукта, а его предложения и улучшения отрабатываются в следующем спринте. Тестирование проводится в каждом спринте, чтобы минимизировать риск любых неудач.

Agile - это философия с ценностями и принципами, но не с жесткими условиями построения процессов.

Основные ценности:

1.Личности и взаимодействие вместо процессов и инструментов.

2.Рабочее программное обеспечение вместо исчерпывающей документации.

3.Сотрудничество с клиентами вместо переговоров по контракту.

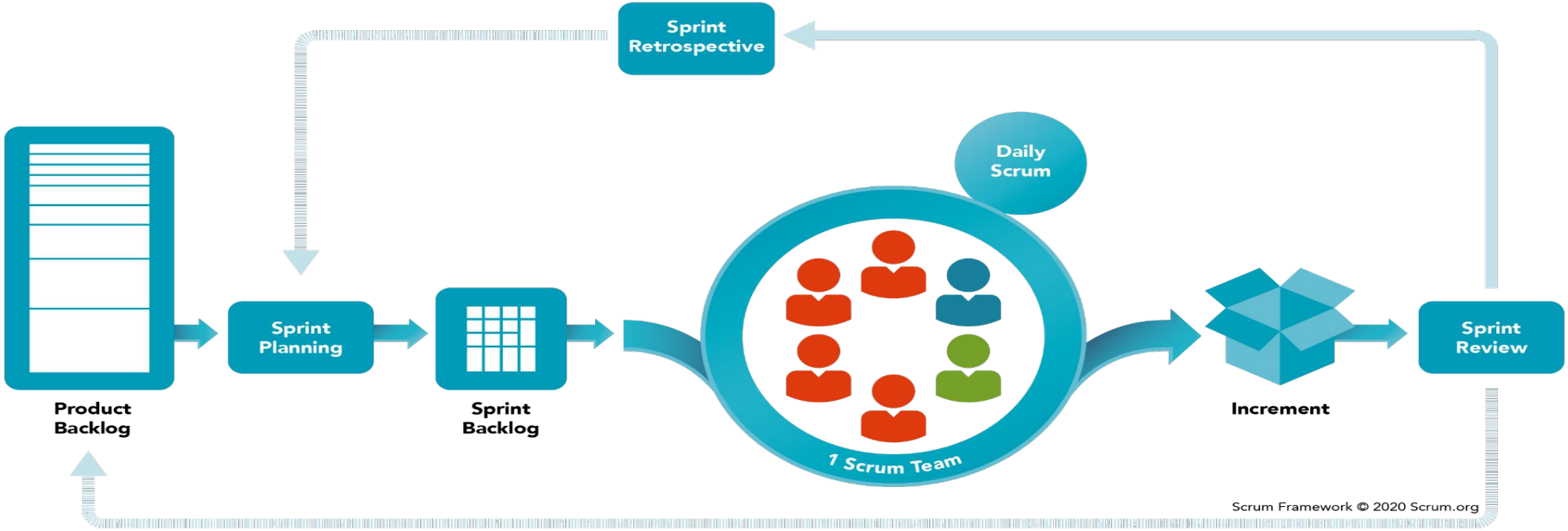
4.Реагирование на изменения вместо следования плану.

Еще существует 12 принципов Agile манифеста

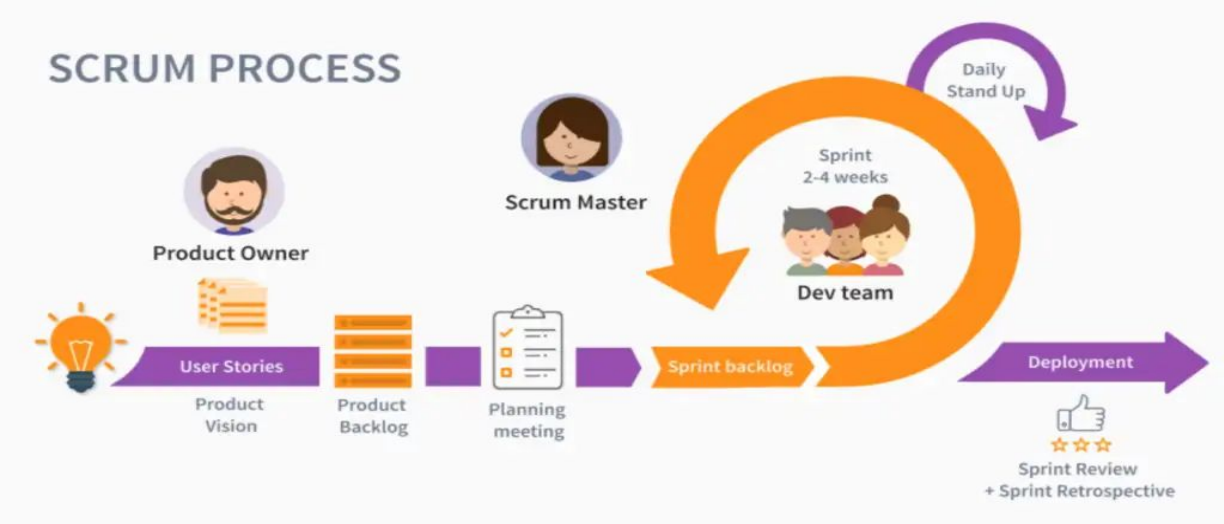
Scrum

Scrum это фреймворк с набором определенных церемоний и ролей, который предназначен для предоставления ценности заказчику на протяжении всего развития проекта.

Метод Scrum - это адаптируемая, быстрая, гибкая и эффективная agile-система. Основной целью Scrum является удовлетворение потребностей заказчика через среду прозрачного общения, коллективной ответственности и непрерывного прогресса.



Scrum роли,церемонии, артефакты



Основные артефакты:
Product backlog (Бэклог продукта)
Sprint backlog (бэклог спринта)
Product increment (инкремент продукта)

Выделяют также метрики:
Sprint burndown chart (график выполнения спринта)
Sprint goal (цели спринта)
Team capacity
Team velocity

Основные роли:

Product owner (Владелец продукта)

Scrum Master (Скрам Мастер)

Development Team (команда разработки, включая QA, BA, etc)

Основные церемонии:

1.Спринт груминг (Sprint grooming/backlog refinement)
2.Планирование спринта (Sprint planning)
3. Ежедневные встречи (Daily Stand-Up)
4. Демонстрация инкремента (Sprint review/Demo)
5. Ретроспектива (Sprint retrospective)

Scrum **роли**: Product owner

Владелец продукта (Product Owner) - это член команды Agile, который выступает в качестве представителя клиента.

Отвечает за работу со Скрам-мастером и другими заинтересованными сторонами по определению и расстановке приоритетов пользовательских историй в бэклоге команды.

Основные обязанности:

Управление и определение приоритетов в бэклоге продукта.

Перевод стратегий клиентов в задачи разработки.

Анализ рынка и потребностей клиентов.

Выполнение функций связующего звена между продуктом и командой разработчиков.

Поясняет пользовательские истории команде .

Scrum роли: Scrum master, Development team

Скрам мастер несет основную ответственность за то, чтобы все понимали и принимали ценности, принципы и практику Scrum.

Необходимо убедиться, что Скрам-мастер правильно понимает Scrum, чтобы Скрам-команда могла воспользоваться его помощью в реализации Scrum правильным способом.

Эта роль не имеет фактических полномочий.

Основные обязанности:

Наставничество членов команды.

Помогать Scrum-команде сосредоточиться на создании высокоценных инкрементов, отвечающих ожиданиям конечных пользователей.

Обеспечение проведения и соблюдения сроков всех мероприятий

Команда разработчиков состоит из инженеров-разработчиков и тестировщиков ПО, которые в конце каждого спринта предоставляют потенциально реализуемый "готовый" прирост продукта (инкремент).

Scrum предписывает, что идеальный размер команды составляет от 3 до 9 членов команды включительно.

Основные обязанности:
Подготовка sprint бэклога продукта.

Планирование спринта.

Выполнение спринта.

Мониторинг хода спринта.

Адаптация продукта и процесса.

Основные церемонии Scrum

1.Sprint Grooming. Обычно проводится до старта спринта. На данном этапе команда уточняет требования, задает вопросы Product Owner, определяет как пользовательские истории могут быть разработаны и протестированы.

2.Sprint planning. Обычно проводится в первый день спринта. На данном этапе оценивается работа, которая может быть выполнена путем приоритезации сторей в бэклоге, выбираются стори для спринта, пользовательские истории разбиваются на задачи, для каждой стори/задачи проводится эстимация (оценка трудозатрат).

3. Daily StandUp/Daily scrum. Обычно проводится Скрам мастером. Каждый член команды докладывает: что сделано, что планируется делать, какие трудности существуют. Длительность такой встречи занимает 10-15 минут.

4. Sprint review/Demo. Обычно проводится в конце спринта, когда запланированный пул задач выполнен. Один из членов команды демонстрирует заказчику выполненную работу (разработанный функционал). Команда получает фидбэк от клиента.

5. Sprint retrospective. Обычно проводится в начале следующего спринта либо в конце текущего спринта. На данном этапе команда обсуждает предыдущий спринт: что было сделано хорошо, что можно улучшить, определяет практические меры (action items) для улучшения процессов в последующих спринтах

Артефакты SCRUM

Product backlog (Бэклог продукта) - это список новых функций, исправлений ошибок, улучшений, изменений в существующей функциональности продукта, которые должны быть реализованы в рамках проекта или разработки продукта.

Sprint backlog (бэклог спринта) - это список задач/ сторей, определенных командой Scrum, которые должны быть выполнены в течение спринта Scrum.

Product increment (инкремент продукта) - это небольшая часть конечного продукта, имеющая свою ценность. С каждым спринтом прирост продукта увеличивается с точки зрения поставляемой функциональности.

Выделяют также:

Sprint burndown chart (график выполнения спринта) - показывает объем работы, который был выполнен в ходе спринта, и общее количество оставшейся работы.

Definition of done (критерии готовности) - это набор критериев, которые должны быть выполнены, чтобы проект считался "выполненным".

Sprint goal (цели спринта) - позволяет всем членам команды понять, что должно быть достигнуто в течение спринта

User story (Пользовательские истории)- общее объяснение функции программного обеспечения, написанное с точки зрения конечного пользователя или клиента.

Kanban

Слово Kanban имеет японское происхождение, и его значение связано с концепцией "точно в срок". На практике метод Канбан организуется на доске или таблице (доска Канбан), разделенной на столбцы, на которых отображается каждый поток в рамках проекта по производству программного обеспечения. По мере развития разработки информация, содержащаяся в таблице, меняется, и каждый раз, когда появляется новая задача, создается новая "карточка".

Метод Kanban требует коммуникации и прозрачности, чтобы члены любой команды точно знали, на какой стадии находится разработка, и могли в любой момент увидеть статус проекта.

Метод Канбан вращается вокруг доски Канбан. Это инструмент, который визуализирует весь проект для отслеживания его течения. Благодаря такому графическому подходу досок Канбан новый участник или внешний субъект может понять, что происходит прямо сейчас, выполненные задачи и будущие задачи.

Канбан-доска показывает (пример):

- текущие задачи, которые выполняются (Doing/in Progress)
- задачи, которые необходимо выполнить в будущем (To Do)
- выполненные задачи (Done)

Разделенные столбцы взаимосвязаны, и задачи постепенно перемещаются из крайнего левого столбца (будущие задачи) в крайний правый столбец (выполненные задачи).

Система Канбан измеряет завершенность рабочего цикла через принцип незавершенного производства (Work in Progress). WIP устанавливают максимальный объем работы, который может существовать в каждом статусе рабочего процесса.

Ограничение WIP для поддержания постоянных стандартов является одним из основных принципов, на которых базируется методология Kanban в Agile. Для команды чрезвычайно важно выполнять текущие задачи в установленном порядке.

Канбан фокусируется на разбиении работы на небольшие задачи, их визуализации и получении небольшого количества элементов в любом рабочем состоянии.

В Канбан также нет предписанных ролей, и ни один человек не несет ответственности за команду или задачу

Kanban подходит для таких проектов как поддержка продукта; проектов, которые имеют много изменений в бэклоге в процессе спринта; которые не имеют четко описанных сторей

Канбан-команда сосредоточена только на работе, которая активно выполняется. Как только команда завершает рабочий элемент (work item), они берут следующий рабочий элемент из списка невыполненных работ(backlog).

Владелец продукта (Product owner) может изменить приоритеты работы в невыполненной работе, не нарушая работу команды, поскольку любые изменения, выходящие за рамки текущих рабочих элементов, не влияют на команду.

Пока владелец продукта держит самые важные рабочие элементы в списке невыполненных работ, команда разработчиков может быть уверена, что они принесут максимальную пользу бизнесу. Таким образом, нет необходимости в итерациях фиксированной длины, как в Scrum.

Канбан церемонии:

- Daily Kanban
- Replenishment meeting
- Service Delivery Review
- Operations Review (applicable mostly for high-level management)
- Strategy Review (applicable mostly for high-level management)

https://kanbanize.com/agile/project-management/ceremonies

Extreme programming (XP)(according to Kent Beck)

Экстремальное программирование (XP) - это гибкая система разработки программного обеспечения, целью которой является создание более качественного программного обеспечения в короткие сроки.

Некоторые выжимки из основных принципов:

- Маленькие релизы
- Простой дизайн (девиз “ничего лишнего”)
- Тестирование: юнит тесты и автотесты
- Рефакторинг (улучшение кода без его изменения)
- Парное программирование (один пишет код, второй смотрит и исправляет)
- Коллективное владение (любой программист может изменять любую часть кода)
- Непрерывная интеграция
- 40-часовая неделя
- Постоянная обратная связь от клиента
- Стандарт кодирования

По словам Кента Бека, зрелая команда XP не должна полагаться на жесткие роли, но тем не менее, есть несколько общих ролей, которые связывают с XP:

Клиент: в идеале должен присутствовать реальный клиент, который будет отвечать на вопросы, определять приоритеты пользовательских историй или участвовать в приемочном тестировании. Когда это невозможно, эту роль может выполнять представитель заказчика.

Программисты: в команде XP программисты оценивают усилия, необходимые для выполнения задач и историй, написания автоматизированных тестов и реализации историй.

Тренер: Наличие тренера не обязательно, но наличие кого-то опытного в XP для обучения команды может гарантировать, что члены команды будут следовать практикам, превращать их в привычки и не возвращаться к старым методам.

Трекер: Трекер отслеживает показатели прогресса команды и разговаривает с каждым членом команды, чтобы выявить препятствия и найти решения. Трекер вычисляет метрики, показывающие, насколько хорошо работает команда, либо для этого используют тулы, где вся статистика рассчитывается автоматически (Джира и тд)

Они делятся на три основные группы: разработка программного обеспечения (software engineering), рабочая среда(work environment) и управление проектами (project management).

Software Engineering:

Парное программирование: в XP вы пишете код парами, сидя за одной машиной. Вы и ваша пара общаетесь друг с другом, анализируя, внедряя и тестируя функцию, над которой работаете. Парное программирование особенно эффективно для создания кода с меньшим количеством дефектов, а также увлекательно, весело и утомительно.

Сборка за десять минут. Ожидается, что сервер непрерывной интеграции создаст весь проект, включая запуск всех автоматических тестов, максимум за десять минут. Это ограничение служит для того, чтобы тесты были сфокусированными. Программирование с началом тестирования: в XP вы реализуете свои функции, используя подход «сначала тестирование» (TDD - это будет домашка)

Work environment:

в XP команды предпочитают работать вместе на открытом пространстве. Эта практика способствует общению и чувству принадлежности к команде.

Информативное рабочее пространство использует физическое пространство команды для отображения информации, которая позволяет любому с первого взгляда узнать о ходе проекта. То, как это делается, может варьироваться: от физических заметок и диаграмм до экранов, отображающих доски Канбан и информационные панели из программного обеспечения для управления проектами.

Энергичная работа: в XP вы работаете только до тех пор, пока можете выполнять энергичную работу. Рабочее время должно быть ограничено максимум 40 часами в неделю.

Project management:

Требования пишутся в формате пользовательских историй (User story). Пользовательская история имеет короткое описательное имя, а также краткое описание того, что должно быть реализовано. Всегда можно добавить больше историй, если команда перевыполнит задание либо отбросить второстепенные по приоритету задачи.

Разработка в XP работает в двух основных циклах: недельном цикле и квартальном цикле. Первоначально Кент Бек рекомендовал двухнедельный цикл, но изменил его во втором издании своей книги.

Недельный цикл: Недельный цикл — это «пульс» XP-проекта. Цикл начинается со встречи, на которой заказчик выбирает, какие истории он хочет реализовать в течение недели. Кроме того, команда анализирует свою работу, включая прогресс за последнюю неделю, и думает, как улучшить свой процесс.

Домашнее задание: Найти и изучить материалы по следующим темам

1) Three amigos session: what is it ?
2) TDD and BDD. Can these things be used together?
3) Gherkin language: what is it ?

The End for today