Контрольная работа

1. Прислать задания по UML-диаграммам (<https://github.com/Lovehansa046/UML_Diagramm>).

2. Дайте определения понятиям (либо ответьте на вопросы).

1. SCRUM

это легкая, но мощная методология управления проектами, основанная на итеративной разработке и эмпирическом процессе. Она помогает командам работать более эффективно, быстрее реагировать на изменения и создавать продукты, которые отвечают потребностям пользователей.

2. KANBAN

это система управления задачами, которая помогает командам визуализировать свою работу и оптимизировать поток задач. KANBAN основан на принципе "точно вовремя" и помогает командам работать более эффективно, быстрее реагировать на изменения и создавать продукты, которые отвечают потребностям пользователей.

1. Пользовательские истории. Backlog

**Пользовательская история** - это описание функциональности программного обеспечения с точки зрения пользователя. Она представляет собой краткое, понятное и нетехническое описание того, что пользователь хочет делать с системой.

4. Водопадная модель

**Водопадная модель** – это линейная методология разработки программного обеспечения, в которой весь процесс разработки делится на последовательные этапы, выполняемые один за другим.

5. Преимущества и недостатки водопадной модели Преимущества водопадной модели:

* Простота: Водопадная модель проста в понимании и использовании.
* Предсказуемость: Водопадная модель позволяет легко предсказать сроки и бюджет разработки.
* Контроль: Водопадная модель обеспечивает четкий контроль над процессом разработки.

Недостатки водопадной модели:

* Негибкость: Водопадная модель негибка и не позволяет легко менять требования к программному обеспечению.
* Риск ошибок: Водопадная модель может привести к появлению ошибок, которые будут обнаружены только на поздних этапах разработки.
* Несоответствие ожиданиям: Водопадная модель может привести к тому, что конечный продукт не будет соответствовать ожиданиям пользователей.

## 6. Преимущества и недостатки Agile методик Преимущества Agile методик:

* Гибкость: Agile-методики позволяют легко адаптироваться к меняющимся требованиям и приоритетам.
* Быстрое реагирование на изменения: Agile-методики позволяют быстро реагировать на изменения рынка и потребности пользователей.
* Повышение качества: Agile-методики помогают повысить качество программного обеспечения за счет постоянного тестирования и обратной связи.
* Улучшение коммуникации: Agile-методики стимулируют улучшение коммуникации между командами и клиентами.
* Повышение мотивации: Agile-методики повышают мотивацию команды за счет вовлечения в процесс разработки и прозрачности.

Недостатки Agile методик:

* Сложность: Agile-методики могут быть сложными для понимания и использования, особенно для начинающих команд.
* Требует дисциплины: Agile-методики требуют от команды дисциплины и самоорганизации.
* Не подходит для всех проектов: Agile-методики не подходят для всех проектов, например, для проектов с фиксированными сроками или бюджетами.
* Риск ошибок: Agile-методики могут привести к появлению ошибок, которые будут обнаружены только на поздних этапах разработки.
* Несоответствие ожиданиям: Agile-методики могут привести к тому, что конечный продукт не будет соответствовать ожиданиям пользователей.

7. Кто такие stakeholders в Agile методике? **Stakeholders** (заинтересованные стороны) – это люди или группы людей, которые могут повлиять на проект или быть затронуты его результатами. В Agile-методике stakeholders играют важную роль, так как их участие помогает команде лучше понять потребности рынка, улучшить качество продукта и повысить шансы на успех проекта.

8. Кто такой Scrum-мастер, какая у него роль. Scrum-мастер – это ключевая роль в Agile-методологии Scrum, направленная на помощь команде в эффективной работе и достижении целей проекта.

9. Как можно определить - сколько пользовательских историй может выполнять команда разработчиков Факторы, влияющие на пропускную способность команды:

* Размер команды: Чем больше команда, тем больше пользовательских историй она может выполнить.
* Опыт команды: Опытные команды могут выполнять больше пользовательских историй, чем команды с меньшим опытом.
* Сложность пользовательских историй: Сложные пользовательские истории требуют больше времени и усилий, чем простые.
* Наличие инструментов и инфраструктуры: Наличие необходимых инструментов и инфраструктуры может повысить пропускную способность команды.
* Процесс разработки: Эффективный процесс разработки может повысить пропускную способность команды.
* Внешние факторы: Внешние факторы, такие как изменения требований или отвлекающие факторы, могут снизить пропускную способность команды.

## 10. Как появилась методология Agile Зарождение Agile-методологии:

1960-е:

* Ранние идеи:
  + Лесли Лампорт: "Декомпозиция задач на итерации" (1968)
  + Уэйн Ройс: "Водопадная модель" (1970)

1980-е:

* Методологии "легкого" программирования:
  + Экстремальное программирование (XP) (Кент Бек, 1996)
  + Crystal Methods: Alistair Cockburn (1991)
  + Feature Driven Development (FDD): Jeff De Luca (1994)

1990-е:

* Развитие Scrum:
  + Ken Schwaber: "Scrum Development Process" (1995)
  + Jeff Sutherland: "Scrum: The Art of Doing Twice the Work in Half the Time" (1998)

2001:

* Agile Manifesto:
  + 17 разработчиков ПО собрались на лыжном курорте Snowbird, штат Юта
  + Опубликовали "Манифест гибкой разработки программного обеспечения"
  + 4 ценности и 12 принципов Agile-подхода

2000-е - настоящее время:

* Распространение Agile:
  + Различные Agile-фреймворки: Scrum, Kanban, Lean, Scrumban
  + Agile применяется не только в разработке ПО, но и в других сферах

Ключевые события в истории Agile:

* 1960-е: Ранние идеи итеративного развития
* 1980-е: Появление методологий "легкого" программирования
* 1990-е: Развитие Scrum
* 2001: Публикация Agile Manifesto
* 2000-е - настоящее время: Распространение Agile

Важные личности в истории Agile:

* Кент Бек: Экстремальное программирование (XP)
* Alistair Cockburn: Crystal Methods
* Jeff De Luca: Feature Driven Development (FDD)
* Ken Schwaber: Scrum
* Jeff Sutherland: Scrum
* Авторы Agile Manifesto: 17 разработчиков ПО

11. Манифест Agile

Манифест Agile

Мы ценим:

* Людей и их взаимодействие больше, чем процессы и инструменты.
* Рабочий продукт больше, чем исчерпывающую документацию.
* Сотрудничество с клиентом больше, чем согласование условий.
* Адаптивность к изменениям больше, чем следование плану

12. Валидация и верификация. В чем разница?

Ключевые отличия:

* Цель: Верификация фокусируется на соответствии спецификациям, валидация – на соответствии ожиданиям.
* Момент: Верификация проводится во время разработки, валидация – после завершения разработки.
* Методы: Верификация использует формальные методы (анализ кода, статическое тестирование), валидация – эмпирические методы (тестирование с участием пользователей).
* Ответственность: Верификация – ответственность команды разработки, валидация – ответственность команды тестирования.

13. Кто такой product-owner?

Product Owner (Владелец продукта) – это ключевая роль в Agile-методологиях, особенно в Scrum.

Основные задачи Product Owner-а:

* Формирование видения продукта: Product Owner должен иметь чёткое видение того, каким должен быть продукт, и уметь донести его до команды.
* Управление Product Backlog-ом: Product Owner отвечает за создание, приоритизацию и управление Product Backlog-ом, который представляет собой список задач, необходимых для разработки продукта.
* Приоритизация задач: Product Owner должен расставлять приоритеты задач в Product Backlog-е, основываясь на ценности для бизнеса, срочности и других факторах.
* Сотрудничество с командой: Product Owner тесно сотрудничает с командой разработчиков, чтобы помочь им понять задачи и цели проекта.
* Принятие решений: Product Owner принимает решения о том, какие функции будут включены в продукт, а какие – нет.
* Общение с stakeholders: Product Owner общается с stakeholders (заинтересованными сторонами), чтобы узнать их потребности и ожидания.

14. Какие методики существует в рамках Agile?

Методики Agile:

В рамках Agile существует множество методик, но наиболее популярными являются:

* Scrum:
  + Итеративная методика, основанная на коротких циклах разработки (спринтах).
  + Включает в себя такие артефакты, как Product Backlog, Sprint Backlog, Sprint Planning, Daily Scrum, Sprint Review, Sprint Retrospective.
* Kanban:
  + Визуальный метод управления потоком задач.
  + Использует доску Kanban, на которой задачи представлены карточками.
  + Ограничивает количество задач, находящихся в работе одновременно (WIP).
* Lean:
  + Фокусируется на минимизации waste (потерь) и оптимизации workflow (потока работ).
  + Использует такие принципы, как kaizen (непрерывное совершенствование), jidoka (автоматизация) и pull (вытягивание).
* Extreme Programming (XP):
  + Сосредоточена на feedback (обратной связи), collaboration (сотрудничестве) и continuous improvement (непрерывном совершенствовании).
  + Включает в себя такие практики, как pair programming (парное программирование), refactoring (рефакторинг) и test-driven development (разработка, управляемая тестированием).
* Crystal Methods:
  + Семейство Agile-методологий, ориентированных на людей и их взаимодействие.
  + Подчеркивает важность empowerment (расширения прав и возможностей) команды, communication (коммуникации) и reflection (рефлексии).

15. Что такое «спринт» в Scrum

Спринт – это фиксированный по времени период разработки в Scrum, который обычно длится от 1 до 4 недель.

16. Что обсуждается на Scrum meeting. Какие вопросы обычно обсуждаются?

Что обсуждается на Scrum meeting (Daily Scrum)

Daily Scrum – это короткая ежедневная встреча, которая проводится командой Scrum.

Цель Daily Scrum:

* Синхронизация работы команды.
* Обнаружение и решение проблем.
* Оценка прогресса в достижении цели спринта.

На Daily Scrum обычно обсуждаются следующие вопросы:

* Что я сделал(а) вчера?
* Что я буду делать сегодня?
* Есть ли у меня какие-либо препятствия?

17. Что такое разработка программного обеспечения

## Разработка программного обеспечения (ПО) – это комплекс мер, направленный на создание нового ПО или же на доработку/усовершенствование уже существующего.

18. Инкрементальная модель

## **Инкрементная модель** – это модель разработки программного обеспечения (ПО), в которой ПО разрабатывается и поставляется поэтапно, добавляя функциональность и улучшения на каждом этапе.

19. Как оценивается сложность выполнения пользовательской истории

Существует несколько методов оценки сложности выполнения пользовательской истории:

* Оценка по очкам истории:
  + Каждой пользовательской истории присваивается определенное количество очков, которое отражает ее сложность.
* Планирование покеру:
  + Члены команды оценивают сложность пользовательской истории, используя колоду карт с очками.
* Оценка по размеру:
  + Сложность пользовательской истории оценивается по ее размеру, например, по количеству строк кода или по количеству функций.
* Оценка по аналогии:
  + Сложность пользовательской истории оценивается по сложности аналогичных пользовательских историй, которые были реализованы ранее.

20. Приведите примеры, где Agile-методика не применима.

1. Проекты с жёстко фиксированными требованиями:

* Разработка медицинского оборудования:
  + Требования к безопасности и функциональности strictly regulated (строго регламентированы).
* Создание космических аппаратов:
  + Невозможно iteratively develop (итеративно разрабатывать) космический аппарат due to the high cost of failure (из-за высокой стоимости ошибки).

2. Проекты с низким уровнем сложности:

* Разработка простых веб-сайтов:
  + Agile-методология может быть overkill (избыточной).
* Создание небольших программ:
  + Agile-методология может add unnecessary overhead (добавить ненужные накладные расходы).

3. Проекты с ограниченным бюджетом:

* Agile-методология может быть more expensive (более дорогой), чем традиционные методы:
  + Требует dedicated Scrum Master (специалиста по Scrum) and Agile Coach (коуча Agile).
* Agile-методология может lead to scope creep (привести к разрастанию функциональности):
  + Если not managed properly (не управлять должным образом), new features (новые функции) могут быть added (добавлены) unnecessarily (необоснованно).

4. Проекты с распределенной командой:

* Agile-методология relies on communication and collaboration (основана на общении и сотрудничестве):
  + Difficult to achieve (трудно достичь) with a distributed team.
* Cultural differences (культурные различия) can make it difficult to implement Agile practices (внедрить Agile-практики).

5. Проекты с короткими сроками:

* Agile-методология requires time for iteration and feedback (требует времени для итераций и обратной связи):
  + May not be enough time (может не хватить времени) if the project has a short deadline.
* Agile-методология can be disruptive (может быть деструктивной):
  + Frequent changes (частые изменения) can be problematic (проблематичными) if the project is on a tight schedule (сжатые сроки).

21. Определение «проекта»

**Проект** – это целенаправленная, ограниченная во времени деятельность, осуществляемая для достижения конкретных целей при наличии внешних и внутренних ограничений и использовании ограниченных ресурсов.

22. Зачем нужно техническое задание?

Оно является важным инструментом для:

* Заказчика:
  + Позволяет clearly formulate (четко сформулировать) свои требования and expectations (и ожидания).
* Исполнителя:
  + Позволяет clearly understand (четко понять) задачу and develop (разработать) продукт, который соответствует требованиям заказчика.

23. Главное отличие коммерческой организации от некоммерческой.

Главное отличие коммерческой организации от некоммерческой:

Цель деятельности:

* Коммерческая организация:
  + Главная цель - получение прибыли.
  + Прибыль распределяется между участниками или используется для развития организации.
* Некоммерческая организация:
  + Главная цель - не получение прибыли, а достижение социальных, культурных, образовательных или других целей.
  + Прибыль не распределяется между участниками, а используется для достижения целей организации.

24. Какая стратегия является приоритетной: бизнес-стратегия или ИТ-стратегия в организации?

* Бизнес-стратегия имеет приоритет над ИТ-стратегией:
  + ИТ-стратегия должна быть aligned with (согласована с) бизнес-стратегией.
* ИТ-стратегия является enabler (инструментом) для реализации бизнес-стратегии:
  + ИТ-решения should help (должны помогать) организации achieve its goals (достигать своих целей).

25. Назовите одну-две категории ИТ-проектов.

**1. Разработка программного обеспечения**

**2. Внедрение ИТ-инфраструктуры**

26. Назовите один-два варианты специализации кадров в ИТ-секторе

**Специалист по информационной безопасности**

**Web-разработчик**

27. Назовите один из принципов в системе управлении временем

* Расстановка приоритетов:
  + Определите, какие задачи являются наиболее важными и срочными.

28. Какие существуют процессы управления сроками?

**1. Инициация и определение**

**2. Планирование**

**3. Выполнение**

**4. Контроль и мониторинг**

**5. Закрытие**

29. Приведите пример использования диаграммы Ганта.

**1. Определение задач**

**2. Определение этапов**

**3. Оценка длительности задач**

**4. Разработка календарного плана**

**5. Диаграмма Ганта**

30. Определение «риск проекта»

**Риск проекта** - это неопределенное событие или условие, которое может иметь как **положительное**, так и **отрицательное** влияние на **цели проекта**.

31. Необходимые компетенции для Product Manager (PM)

1. Технические навыки:

* Понимание принципов работы программного обеспечения.
* Умение работать с данными.
* Умение создавать прототипы.
* Знание основ UX/UI дизайна.

2. Бизнес-навыки:

* Понимание бизнес-моделей.
* Умение проводить конкурентный анализ.
* Умение управлять продуктом.
* Умение работать с клиентами.

3. Мягкие навыки:

* Аналитическое мышление.
* Коммуникабельность.
* Лидерские качества.
* Креативность.
* Умение работать в команде.

4. Специфические навыки:

* Умение работать с Agile-методологиями.
* Знание принципов управления продуктом.
* Опыт работы в продуктовой команде.

32. Для чего нужна RACI матрица

## **RACI-матрица** (от англ. Responsible, Accountable, Consulted, Informed) – это инструмент, который помогает **определить роли и обязанности** участников проекта.

33. Разница между Kanban и Scrum

Основные отличия:

1. Фокус:

* Kanban:
  + Фокусируется на визуализации рабочего процесса и ограничении незавершенной работы.
* Scrum:
  + Фокусируется на итеративной разработке и регулярной доставке результатов.

2. Структура:

* Kanban:
  + Простая и гибкая структура.
  + Нет фиксированных временных рамок или ролей.
* Scrum:
  + Более структурированная методология.
  + Использует фиксированные временные рамки (спринты) и роли (Scrum Master, Product Owner, Development Team).

3. Планирование:

* Kanban:
  + Непрерывное планирование.
  + Задачи добавляются в канбан-доску по мере необходимости.
* Scrum:
  + Планирование на основе спринтов.
  + Задачи определяются в начале каждого спринта.

4. Изменения:

* Kanban:
  + Легко адаптируется к изменениям.
  + Задачи могут быть добавлены или удалены в любой момент.
* Scrum:
  + Изменения вносятся в конце каждого спринта.
  + Изменения в спринте возможны, но могут быть трудоемкими.

5. Подходит для:

* Kanban:
  + Команд, которые работают над непрерывными потоками задач.
  + Команд, которым нужна гибкость и возможность быстро реагировать на изменения.
* Scrum:
  + Команд, которые работают над проектами с четко определенными целями и сроками.
  + Команд, которые хотят использовать преимущества итеративной разработки.

34. Для чего необходимо VCS для проекта и какие стратегии используются

VCS используется для:

* Сохранения истории изменений проекта.
* Отслеживания изменений, внесенных в файлы.
* Восстановления предыдущих версий файлов.
* Сотрудничества над проектом с другими людьми.

35. Распространенные ошибки у команды разработчиков (developers) в процессе реализации проекта

**1. Нечеткое планирование**

**2. Неэффективная коммуникация**

**3. Неверный выбор инструментов и технологий**

**4. Недостаточное тестирование**

**5. Неумение работать с ошибками**

**6. Неумение работать в команде**

**7. Несоблюдение сроков**

**8. Недооценка важности документации**

**9. Неспособность реагировать на изменения**

**10. Неготовность к выходу на рынок**

36. Чем занимаются Devops

DevOps-инженеры:

* Автоматизируют задачи разработки, тестирования и эксплуатации.
* Создают инфраструктуру, которая позволяет быстро и надежно развертывать программное обеспечение.
* Обеспечивают бесперебойную работу программного обеспечения.

37. Какие признаки хорошего качественного программного кода в проекте?

**1. Читаемость**

**2. Понятность**

**3. Тестируемость**

**4. Модульность**

**5. Производительность**

**6. Безопасность**

**7. Поддерживаемость**

**8. Соответствие стандартам**

**9. Доступность**

38. Что такое SWOT-анализ

**SWOT-анализ** – это метод стратегического планирования, который заключается в выявлении **сильных и слабых сторон**, **возможностей и угроз**, связанных с вашей компанией, продуктом или проектом.