1.  Концепция тестирования. Тестирование как способ обеспечения качества программного обеспечения.

Концепция тестирования - это общая стратегия и подход к тестированию программного обеспечения, которая определяет, каким образом будет проводиться тестирование, какие задачи оно должно решать и какие результаты оно должно получить.

Тестирование является ключевым этапом в разработке программного обеспечения и способом обеспечения его качества. Правильно спланированное и проведенное тестирование помогает выявить и устранить ошибки и проблемы до выпуска программы, что повышает доверие пользователей к продукту.  
2. Тестирование, верификация и валидация – различия в понятиях.

Тестирование - это процесс, в ходе которого выполняются определенные тестовые сценарии или операции с целью оценки работы системы, выявление ошибок, дефектов и отклонений от требований. Оно включает в себя запуск программы с целью проверки ее функциональности, производительности, безопасности и других аспектов.

Верификация - это процесс проверки, основанный на анализе и оценке документации, спецификаций, моделей и других средств, которые определяют требования к системе или продукту. Проводится на ранних этапах разработки и включает в себя анализ и проверку документации, моделей и кода программы.

Валидация - это процесс оценки системы или продукта на стадии его разработки или в результате тестирования. Она проводится на более поздних стадиях разработки и включает в себя проверку программного обеспечения на соответствие функциональным и нефункциональным требованиям, а также на его пригодность к использованию.  
3. Цели и задачи процесса тестирования.

* Выявление ошибок и проблем
* Улучшение качества продукта
* Проверка соответствия требованиям
* Оптимизация процесса разработки

4. Жизненный цикл тестирования.

Жизненный цикл тестирования - это процесс, состоящий из различных этапов, которые выполняются для обеспечения качества программного обеспечения.

Этапы:

* Планирование тестирования
* Анализ требований
* Создание тестовых случаев
* Подготовка среды тестирования
* Выполнение тестов
* Анализ результатов тестирования
* Отчетность
* Устранение дефектов
* Завершение тестирования
* Обратная связь и улучшение

5. Процессы тестирования при разработке программного обеспечения.

* Планирование тестирования
* Разработка тестовых случаев
* Выполнение тестирования
* Анализ результатов тестирования
* Регрессионное тестирование
* Управление ошибками
* Формирование отчетов
* Документирование

6. Техники тестирования требований.

* Разбор требований
* Техника проверки достижимости
* Техника уточнения требований
* Техника трассировки требований
* Техника функционального тестирования
* Техника тестирования производительности
* Техника тестирования безопасности
* Техника тестирования совместимости
* Техника тестирования масштабируемости
* Техника тестирования надежности

7. Чек-листы, тест – кейсы, наборы тест – кейсов.

Чек-листы представляют собой список пунктов, которые необходимо проверить при тестировании. Они могут включать такие элементы, как функциональные требования, негативные сценарии, проверку внешнего вида или совместимости. Чек-листы помогают не упустить важные аспекты тестирования и задокументировать прогресс работы.

Тест-кейсы - это документы, описывающие шаги, которые должны быть выполнены для проведения конкретного теста. Они содержат информацию о входных данных, ожидаемом результате и ожидаемом поведении системы. Тест-кейсы обычно создаются на основе требований к программному обеспечению и помогают проверить его соответствие этим требованиям.

Наборы тест-кейсов представляют собой группу связанных тест-кейсов, которые могут быть выполнены последовательно или параллельно. Наборы тест-кейсов обычно создаются для проверки различных аспектов функциональности или для выполнения тестирования определенного модуля или компонента.  
8. Анализ и тестирование требований

Анализ требований - это процесс изучения и понимания требований, предъявляемых к программному обеспечению.

Тестирование требований - это процесс проверки требований на соответствие их описанию и правильность их реализации.  
9. Организация тестирования. Фазы тестирования, основные проблемы тестирования.

Организация тестирования - это процесс планирования и управления тестированием программного обеспечения.

Фазы тестирования включают следующие этапы: планирование тестирования, проектирование тестов, разработка тестовых случаев, выполнение тестов, анализ результатов и отчетность.

сновные проблемы тестирования могут включать: недостаточное покрытие тестами, неправильное понимание требований, отсутствие достаточных ресурсов для выполнения тестов, недостаточное участие команды разработки, сложность воспроизведения ошибок и решение проблем совместимости.  
10. Тестирование на основе потока управления.

Тестирование на основе потока управления - это метод тестирования, который основывается на проверке различных вариантов исполнения программы в зависимости от переходов в управляющих конструкциях (например, условных операторах или циклах).  
11. Тестирование на основе потока данных. Тестовые сценарии.

Тестирование на основе потока данных - это метод тестирования, который фокусируется на проверке потока данных в программе

Тестовые сценарии - это последовательности действий, которые выполняются в процессе тестирования для проверки определенного аспекта программы или функции.  
12. Критерии выбора тестов. Требования к идеальному критерию тестирования и классы частых критериев.

Критерии выбора тестов - это правила или метрики, по которым тестировщик определяет, какие тесты следует выполнить для достижения наилучшей эффективности в процессе тестирования.

Критерии:

* Высокое покрытие кода программы или функции.
* Достаточная способность обнаружить ошибки.
* Эффективность использования ресурсов (время, усилия и т. д.).
* Понятность и простота применения.
* Устойчивость к изменениям в программе.

Классы:

* Покрытие кода - оценивает, насколько код программы был протестирован.
* Эквивалентное разбиение - разделение входных данных на группы, где входные данные в каждой группе считаются эквивалентными в отношении выполнения программы.
* Метод граничных значений - фокусируется на тестировании граничных значений входных данных, так как они часто являются источниками ошибок.
* Тестирование ошибок - основывается на опыте и интуиции тестировщика для предсказания возможных ошибок или их источников в программе.

13. Тестирование  программного продукта по структурным критериям

Тестирование программного продукта по структурным критериям - это процесс проверки программного продукта на соответствие его внутренним структурам и логике работы.  
14. Мутационный критерий тестирования.

Мутационный критерий тестирования основан на создании и изменении дефектов в программном коде, чтобы проверить, насколько хорошо тестовые случаи обнаруживают эти дефекты. Для этого используется инструмент, известный как мутационный тестировщик или мутационный средство. Этот инструмент создает измененные версии программы (мутанты), каждый из которых содержит одну или несколько мутаций, такие как изменение операторов или удаление условных выражений. Затем тестировщики запускают тестовые случаи на каждом мутанте и проверяют, обнаруживает ли тестовый набор мутацию. Если тестовый случай не обнаруживает мутацию, это указывает на неполное тестирование и необходимость дополнительных тестов.  
15. Стохастический критерий тестирования.

Стохастический критерий тестирования основан на использовании статистических методов и вероятностных моделей для генерации и выбора тестовых случаев. В этом подходе тестеры используют стохастическое моделирование и случайные процессы, чтобы создать случайные входные данные и проверить, как программный продукт реагирует на них. Использование стохастического критерия позволяет более равномерно иследовать пространство возможных входных данных и идентифицировать потенциальные проблемы в программе, которые могут быть пропущены при других методах тестирования.

16. Тестирование по методу «белого ящика»

Тестирование по методу «белого ящика» - это метод тестирования программного продукта, основанный на анализе его внутренней структуры и логики работы. Тестирование по методу «белого ящика» предполагает доступ к исходному коду программы и позволяет проверить все пути выполнения программы, а также выявить возможные ошибки и дефекты, связанные с ее внутренней логикой. Для тестирования по методу «белого ящика» используются техники, такие как покрытие кода тестами, анализ циклов и условий, тестирование пограничных случаев и т. д.  
17. Тестирование по методу «черного ящика»

Тестирование по методу «черного ящика» - это метод тестирования программного продукта, при котором тестируется только его внешнее поведение без доступа к исходному коду программы. Тестирование по методу «черного ящика» основано на спецификациях требований и функциональности программы. Тестирование по методу «черного ящика» позволяет проверить, соответствует ли программный продукт требованиям, как правильно выполняется каждая функция и как взаимодействует с другими компонентами системы. Для тестирования по методу «черного ящика» используются техники, такие как тестирование эквивалентных классов, граничных значений, случайных величин и т. д.

18. Модульное тестирование - это процесс проверки отдельных компонентов программного обеспечения, называемых модулями, для убедительности в их корректной работе. Основные особенности модульного тестирования включают в себя:

Исключение зависимостей: модули тестируются независимо от других модулей, чтобы изолировать исследуемую функциональность.

Неразрушающие изменения: модульное тестирование не должно изменять или повреждать исходный код или структуру модуля.

Максимальное покрытие: модульные тесты должны проверять все возможные пути выполнения и граничные условия внутри модуля.

Подходы к тестированию на основе потока управления включают тестирование по каждому блоку кода, где каждое возможное ветвление рассматривается отдельно. Тестирование на основе потока данных включает проверку различных наборов данных, которые проходят через модуль, чтобы убедиться, что он правильно обрабатывает их.

Организация модульного тестирования включает определение тестовых случаев и создание тестовых сценариев для каждого модуля, разработку тестовых данных и запуск тестов. Результаты тестирования должны быть записаны и проанализированы для выявления ошибок и исправления их до интеграции модулей.

19. Статические методы при структурном подходе включают использование инструментов и техник анализа кода без его запуска. Это может включать обзор кода, проверку форматирования и применение линтеров. Статические методы помогают выявить потенциальные проблемы на уровне кода до его выполнения.

Динамические методы при структурном подходе включают запуск программного кода и проверку его поведения во время выполнения. Это может включать тестирование на основе потока управления и потока данных, как описано выше.

При объектном подходе статические методы могут быть использованы для анализа структуры классов и их отношений, например, с помощью анализа UML-диаграмм. Динамические методы могут быть использованы для проверки поведения объектов во время выполнения программы.

20. Тестирование по функциональному критерию - это метод тестирования, при котором функциональность программного продукта проверяется на соответствие требованиям и спецификации. Оценивается, как программа выполняет заданные функции и как она реагирует на различные входные данные. Примерами функциональных критериев являются проверка правильности вычислений, обработки входных данных и генерации ожидаемого вывода.

21. Интеграционное тестирование - это процесс проверки взаимодействия и корректной работы компонентов программного обеспечения вместе. Оно предшествует системному тестированию и включает проверку взаимодействия между модулями, передачу данных и обработку ошибок.

22. Сборка модулей и методы интеграционного тестирования связаны между собой, так как интеграционное тестирование проводится для проверки корректной работы взаимодействия между модулями системы. При сборке модулей происходит объединение отдельных частей системы в единое целое. В процессе интеграционного тестирования проверяется связь, взаимодействие и передача данных между модулями системы.

Подходы к интеграционному тестированию:

Монолитный подход - тестирование системы происходит как единое целое, без разделения на модули. Используется при тестировании небольших систем или при отсутствии явного разделения на модули.

Инкрементальный подход - тестирование происходит поэтапно по мере добавления новых модулей. На каждом этапе тестируются только новые модули и их взаимодействие с уже имеющимися модулями системы.

Нисходящий подход - тестирование начинается с верхних уровней системы и постепенно спускается к нижним. Сначала проверяется работа системы в целом, затем модули верхнего уровня, затем модули следующих уровней и т.д.

Восходящий подход - тестирование начинается с нижних уровней системы и постепенно поднимается к верхним. Сначала проверяются самые низкоуровневые модули, затем модули следующих уровней и т.д.

23. Оценка оттестированности проекта – это процесс определения степени, в которой проект был подвергнут тестированию и оценка, насколько полно и эффективно были проведены тесты. Для оценки оттестированности проекта используются различные метрики и методики.

Метрики для оценки оттестированности проекта могут включать количество проведенных тестов, долю выполненных тестов, количество найденных и исправленных ошибок, среднее время исправления ошибок и другие показатели. Метрики помогают оценить степень покрытия системы тестами и эффективность тестирования.

Методика оценки оттестированности проекта может включать анализ тестового покрытия, оценку качества тестовых сценариев, анализ результатов выполнения тестов и другие подходы. Оценка оттестированности проекта позволяет определить, насколько хорошо система прошла тестирование и готова к выпуску.

24. Системное тестирование – это вид тестирования, проводимый на готовом продукте или системе в целом. Целью системного тестирования является проверка работы и взаимодействия компонентов системы, ее соответствия требованиям, устойчивости, производительности и других характеристик системы.

В рамках системного тестирования могут проводиться различные виды тестов, такие как функциональное тестирование, нагрузочное тестирование, тестирование безопасности, тестирование совместимости и другие.

25. Регрессионное тестирование - это процесс проверки программного продукта для выявления новых ошибок после внесения изменений в код или добавления нового функционала. Особенностью регрессионного тестирования является то, что его целью не является проверка всех функций программы, а только тех, которые могут быть затронуты в результате изменений. Виды регрессионного тестирования могут варьироваться в зависимости от регрессионных областей (изменения, которые необходимо проверить) и выбранной стратегии тестирования.

26. Особенности регрессионного тестирования для объектно-ориентированного программирования (ООП) включают проверку внешних и внутренних связей между классами и модулями, проверку изменений в интерфейсах классов и регистрацию нарушений принципов ООП, таких как нарушение инкапсуляции или наследования. Регрессионное тестирование для ООП также требует обновления и адаптации тестового набора для учета изменений в коде, классах и их отношениях, чтобы убедиться, что новые изменения не вызывают регрессионные ошибки.

27. Документирование процесса тестирования включает в себя создание и поддержку документов, которые описывают все шаги, процедуры и результаты тестирования программного продукта. Это может включать в себя планы тестирования, тестовые сценарии, тестовую документацию, отчеты о выполненных тестах и другие связанные документы.

28. Документирование и жизненный цикл дефекта включает в себя процесс регистрации, отслеживания и управления дефектами, найденными в программном продукте во время тестирования или эксплуатации. Жизненный цикл дефекта обычно включает следующие этапы:

Регистрация дефекта: дефект регистрируется в системе управления дефектами, где указываются его описание, приоритет, статус и другие атрибуты.

Отслеживание и исправление дефекта: дефект назначается разработчику для исправления, и его статус отслеживается. Когда дефект исправлен, он проверяется тестировщиком.

Подтверждение исправления дефекта: если дефект исправлен, тестировщик проводит повторное тестирование, чтобы убедиться, что дефект действительно исправлен.

Закрытие дефекта: если дефект более не обнаруживается и считается исправленным, он закрывается в системе управления дефектами.

29. Отчеты о дефектах представляют собой документы, которые содержат информацию о каждом обнаруженном дефекте в программном продукте.

30. Ошибки, дефекты, сбои и отказы - это термины, используемые для описания разных видов проблем, связанных с программными продуктами:

* Ошибки (errors): это некорректные действия или цепочка событий, вызванные человеческим фактором, как например, неправильный ввод данных или некорректное использование функций программы.
* Дефекты (defects): это ошибки или неправильности в программе, которые могут вызывать некорректное поведение или работу программного продукта.

31. Тест-план - это документ, который содержит подробное описание плана тестирования. Он включает в себя информацию о целях, задачах, методах, ресурсах и расписании тестирования.

32. Метрики покрытия требований отражают степень, в которой тесты покрывают определенные требования к системе или продукту. Метрики покрытия кода, с другой стороны, измеряют, насколько код программы был протестирован.

33. Составление отчета о тестировании - это процесс подготовки документа, в котором отражаются результаты проведенного тестирования. В отчете о тестировании обычно содержатся информация о методологии тестирования, описание проведенных тестовых случаев, результаты их выполнения, выводы, рекомендации и другие соответствующие данные.

34. Отчет о дефектах - это документ, который содержит информацию о найденных дефектах в процессе тестирования. Отчет о дефектах имеет свой жизненный цикл, включающий такие этапы, как создание дефекта, его назначение, исправление, верификация и закрытие. Атрибуты или поля отчета о дефектах могут включать информацию о типе дефекта, его приоритете, описании, причине возникновения, статусе, ответственном лице и др.

35. Тестовое покрытие - это мера, которая определяет, насколько полно тесты покрывают функциональность или другие аспекты системы. Высокое тестовое покрытие означает, что большая часть функциональности была протестирована и вероятность обнаружения дефектов высока. Низкое тестовое покрытие может указывать на необходимость проведения дополнительных тестов для обеспечения качества системы.