Вопросы

1. Цели и задачи тестирования, что входит в задачи тестировщика
2. Тестирование пользовательского интерфейса, ручное тестирование интерфейса, плюсы и минусы
3. Тестирование пользовательского интерфейса, автоматическое тестирование интерфейса, плюсы и минусы
4. Участники процесса тестирования, их назначение
5. Жизненный цикл ПО. Этапы.
6. Виды тестирования, интеграционное тестирование, достоинства и недостатки.
7. Виды тестирования, модульное тестирование, достоинства и недостатки.
8. Виды тестирования, системное тестирование, достоинства и недостатки.
9. Функциональное тестирование, дать описание.
10. Инсталляционное тестирование, на что оно направленно.
11. Тестирование безопасности на что оно направленно.
12. Тестирование удобства использования, на что оно направленно.
13. Что такое требование, источники требований, методы выявления требований.
14. Тестовая стратегия, что включает в себя тестовая стратегия.
15. План тестирования, цели плана тестирования.
16. План тестирования, критерии начала и окончания тестирования.
17. Тест-дизайн, цели и задачи тест-дизайна.
18. Тест-кейс, атрибуты тест-кейса, плюсы и минусы тест-кейса.
19. Приоритет и Важность, градации важности дефекта (TestPlan).
20. Модульное тестирование, цели модульного тестирования, драйвер, заглушка.
21. Системное тестирование, объекты выявления при системном тестировании.
22. Интеграционное тестирование, возможное ошибки при интеграции, объекты интеграционного тестирования
23. Что такое тестирование, что такое тест, тестовый случай.
24. Тестовое покрытие, что входит в состав тестового набора, критерии покрытия кода
25. Ошибка, дефект, отказ. Что это такое, в чем отличия.
26. Тестирование пользовательского интерфейса, задачи тестирования пользовательского интерфейса.
27. Верификация и валидация, что это такое, в чем отличия
28. Тестирование пользовательского интерфейса, виды тестирования пользовательского интерфейса.
29. Классы эквивалентности. Анализ граничных значений
30. Матрица принятия решений.
31. Unit тест. Типы тестирования
32. Свойства хорошего Unit Test. Принцип именования Unit Test
33. Когда не нужно создавать Unit Test. Когда нужно создавать Unit Test
34. Шаблон ААА. Методы класса Assert
35. Test Driven Development. Методы класса StringAssert
36. Именование методов и классов тестирования. Методы класса CollectionAssert
37. Тесты состояния. Тесты взаимодействия
38. Внешняя зависимость. Виды тестовых объектов
39. Нагрузочное тестирование. Стресс-тестирование
40. Apache JMeter. Объект Assertion
41. Apache JMeter. Объект Listener
42. Apache JMeter. Объект Samplers

1. Цели и задачи тестирования включают в себя обеспечение соответствия программы её ожидаемому поведению, обнаружение ошибок и дефектов, а также повышение качества программного продукта ,[object Object],. Задачи тестировщика включают в себя анализ требований, разработку тест-кейсов, выполнение тестов, запись ошибок, анализ дефектов и оценку критериев завершения тестирования ,[object Object],.

2. Ручное тестирование пользовательского интерфейса позволяет проверить удобство использования и соответствие дизайна ожиданиям пользователей. Плюсы включают гибкость и способность обнаруживать непредвиденные проблемы, но минусы включают повышенную трудоемкость и невозможность автоматизации ,[object Object],.

3. Автоматическое тестирование пользовательского интерфейса обеспечивает повторяемость, быстроту и возможность интеграции с непрерывной интеграцией и поставкой. Однако минусы включают сложность в создании и поддержке скриптов тестирования, а также невозможность обнаружения некоторых типов проблем ,[object Object],.

4. Участники процесса тестирования включают тест-менеджера, тест-аналитика, тест-дизайнера и тестировщика. Их назначение заключается в планировании, проектировании и выполнении тестов, анализе результатов и создании тестовых случаев ,[object Object],.

5. Жизненный цикл ПО включает этапы разработки, тестирования, доработки и сопровождения. Начало жизни ПО связано с анализом требований и проектированием, а конец жизни - выводом из эксплуатации ,[object Object],.

6. Интеграционное тестирование позволяет проверить взаимодействие между различными модулями или компонентами программы. Достоинства включают обнаружение ошибок взаимодействия, но недостатки - сложность в настройке и поддержке тестов ,[object Object],.

7. Модульное тестирование направлено на проверку отдельных модулей или компонентов программы. Достоинства включают быструю локализацию ошибок, но недостатки - невозможность проверки взаимодействия между модулями ,[object Object],.

8. Системное тестирование позволяет проверить работоспособность всей системы в целом. Достоинства включают проверку конечного продукта, но недостатки - сложность воспроизведения и обнаружения проблем ,[object Object],.

9. Функциональное тестирование направлено на проверку соответствия программы её функциональным требованиям. Это может включать тестирование входных и выходных данных, интерфейсов и процессов ,[object Object],.

10. Инсталляционное тестирование направлено на проверку процесса установки программы, корректности установленных файлов и наличия необходимых зависимостей ,[object Object],.

11. Тестирование безопасности направлено на проверку защищенности программы от внешних угроз, таких как несанкционированный доступ, взломы и утечки данных ,[object Object],.

12. Тестирование удобства использования направлено на оценку удобства, понятности и привлекательности для пользователей разрабатываемого продукта в контексте заданных условий ,[object Object]

13. Требование - это описание того, что должна делать программа или как она должна работать. Источниками требований могут быть заказчик, пользователи, менеджеры проекта и другие заинтересованные стороны. Методы выявления требований включают в себя интервью, опросники, наблюдение, анализ документации и другие ,[object Object],.

14. Тестовая стратегия - это общий план контроля качества ПО, охватывающий длительный период времени. Она включает в себя типы тестов, подходы к тестированию и требования к инструментам и инфраструктуре ,[object Object],.

15. План тестирования - это документ, в котором определены объем, ресурсы и календарный план работ по тестированию. Цели плана тестирования включают понимание того, что, как, когда и кем будет проверяться, а также донесение этой информации до заказчика или команды ,[object Object],.

16. Критерии начала и окончания тестирования могут включать в себя выполнение определенного количества тестов, достижение определенного уровня покрытия кода, устранение всех критических дефектов и другие факторы. План тестирования также может включать в себя описание рисков и меры, которые нужно принимать в случае их возникновения ,[object Object],.

17. Тест-дизайн - это этап процесса тестирования, на котором проектируются и создаются тестовые случаи в соответствии с определенными критериями качества и целями тестирования. Цели тест-дизайна включают придумывание тестов, которые обнаружат наиболее серьезные ошибки продукта, а также минимизацию количества тестов, необходимых для нахождения большинства серьезных ошибок ,[object Object],.

18. Тест-кейс - это документ, описывающий шаги, необходимые для выполнения определенного теста. Атрибуты тест-кейса могут включать в себя идентификатор, описание, шаги теста, ожидаемый результат, фактический результат и статус. Плюсы тест-кейса включают повторяемость и возможность автоматизации, а минусы - сложность в создании и поддержке ,[object Object],.

19. Приоритет и важность - это показатели, используемые для определения того, какие дефекты нужно исправлять в первую очередь. Градации важности дефекта могут включать критические, высокие, средние и низкие. В плане тестирования могут быть определены критерии приоритета и важности для каждого дефекта ,[object Object],.

20. Модульное тестирование - это проверка отдельных модулей или компонентов программы. Цели модульного тестирования включают быструю локализацию ошибок, а также проверку соответствия модуля его спецификации. Для проведения модульного тестирования могут использоваться драйверы и заглушки ,[object Object],.

21. Системное тестирование - это тестирование всей системы в целом, чтобы убедиться, что она соответствует своим требованиям и работает корректно в своем окружении. Объекты выявления при системном тестировании могут включать функциональные и нефункциональные требования, производительность, надежность, безопасность и совместимость ,[object Object],.

22. Интеграционное тестирование - это тестирование взаимодействия между отдельными компонентами системы после их объединения. Возможные ошибки при интеграции включают неправильное взаимодействие между компонентами, неправильную передачу данных и неправильное взаимодействие с внешними системами. Объекты интеграционного тестирования включают интерфейсы между компонентами, передачу данных и взаимодействие с внешними системами ,[object Object],.

23. Тестирование - это процесс проверки программного обеспечения на соответствие его требованиям. Тест - это конкретный сценарий или набор условий, используемых для проверки определенного аспекта программы. Тестовый случай - это документированный сценарий или условия, которые должны быть протестированы ,[object Object],.

24. Тестовое покрытие - это мера, показывающая, насколько программный код был протестирован. В состав тестового набора входят тестовые случаи, которые покрывают различные аспекты программы. Критерии покрытия кода могут включать покрытие строк кода, ветвлений, условий и путей выполнения ,[object Object],.

25. Ошибка - это действие программиста, приводящее к неправильному функционированию программы. Дефект - это несоответствие программы ее спецификации. Отказ - это некорректное поведение программы в ответ на определенные входные данные. Отличие между ними заключается в том, что ошибка - это действие, дефект - его результат, а отказ - проявление дефекта в работе программы ,[object Object],.

26. Тестирование пользовательского интерфейса - это процесс проверки удобства использования, обучаемости, понятности и привлекательности для пользователей разрабатываемого продукта. Задачи тестирования пользовательского интерфейса включают проверку правильности отображения элементов интерфейса, удобство навигации, соответствие дизайну и т.д. ,[object Object],.

27. Верификация - это процесс оценки системы или компонента для убеждения в том, что она была правильно разработана. Валидация - это процесс оценки системы или компонента для убеждения в том, что она соответствует потребностям и ожиданиям пользователя. Отличие между ними заключается в том, что верификация проверяет "правильность", а валидация проверяет "соответствие требованиям" ,[object Object],.

28. Виды тестирования пользовательского интерфейса могут включать функциональное тестирование (проверка функциональности), тестирование удобства использования (проверка удобства для пользователя), тестирование локализации (проверка перевода) и другие ,[object Object],.

29. Классы эквивалентности используются для разделения входных данных на группы, чтобы уменьшить количество тестов. Анализ граничных значений используется для определения тестов на границах разделенных классов эквивалентности ,[object Object],.

30. Матрица принятия решений - это инструмент, используемый для принятия решений о том, какие тесты следует проводить на основе различных факторов, таких как приоритеты, риски и ресурсы ,[object Object],.

31. Unit тест - это тестирование отдельных компонентов (или модулей) программы. Типы тестирования могут включать тестирование функций, классов, методов и модулей ,[object Object],.

32. Свойства хорошего Unit Test включают изолированность (не зависит от других тестов), повторяемость (дает одинаковый результат при повторном запуске), автоматизируемость (может быть запущен без участия пользователя) и другие. Принцип именования Unit Test обычно включает указание тестируемой функциональности и условий ,[object Object],.

33. Необходимость создания Unit Test зависит от конкретной ситуации. Они не нужны, если код неподдающийся тестированию, времени на написание тестов нет, или тесты не приносят пользы. Они нужны, если код критичен для бизнеса, часто меняется, или требуется документация ,[object Object],.

34. Шаблон ААА (Arrange, Act, Assert) - это шаблон для написания тестов, включающий подготовку к выполнению теста (Arrange), выполнение тестируемых действий (Act) и проверку результатов (Assert). Методы класса Assert используются для проверки ожидаемых результатов в тестах ,[object Object],.

35. Test Driven Development (TDD) - это методология разработки, при которой тесты пишутся до написания кода. Методы класса StringAssert используются для проверки строковых значений в тестах ,[object Object],.

36. Именование методов и классов тестирования должно быть информативным и отражать тестируемую функциональность. Методы класса CollectionAssert используются для проверки коллекций в тестах ,[object Object],.

37. Тесты состояния - это тесты, проверяющие поведение системы в различных состояниях. Тесты взаимодействия - это тесты, проверяющие взаимодействие между различными компонентами системы ,[object Object],.

38. Внешняя зависимость - это зависимость компонента от внешних ресурсов или услуг. Виды тестовых объектов могут включать тестирование взаимодействия с базой данных, сетью, файловой системой и другими внешними ресурсами ,[object Object],.

39. Нагрузочное тестирование - это вид тестирования, направленный на проверку работы системы при большой нагрузке, близкой к максимальной. Цель такого тестирования - определить, как система будет работать в реальных условиях, когда на нее будут оказываться высокие нагрузки. Стресс-тестирование - это вид нагрузочного тестирования, направленный на проверку работы системы при экстремальных условиях, когда на нее оказывается максимально возможная нагрузка.

40. Apache JMeter - это инструмент для проведения нагрузочного тестирования. Объект Assertion в Apache JMeter используется для проверки правильности ответа сервера на запросы. Assertion позволяет проверять различные аспекты ответа сервера, такие как код ответа, содержимое ответа, время ответа и т.д. Если Assertion не проходит, то тест считается неудачным.

41. Объект Listener в Apache JMeter используется для отображения результатов тестирования. Listener позволяет просматривать результаты тестирования в различных форматах, таких как таблицы, графики, диаграммы и т.д. С помощью Listener можно анализировать результаты тестирования и выявлять проблемы в работе системы.

42. Объект Samplers в Apache JMeter используется для отправки запросов на сервер во время тестирования. Samplers позволяет отправлять различные типы запросов, такие как HTTP-запросы, FTP-запросы, JDBC-запросы и т.д. С помощью Samplers можно имитировать работу реальных пользователей и проверять работу системы при различных условиях.