**1. Что такое тестирование ПО?**

Тестирование программного обеспечения (ПО) — это систематический процесс проверки, который имеет целью выявление дефектов в программном продукте. Он включает в себя как динамическое, так и статическое тестирование. В ходе тестирования проверяется соответствие ПО заданным требованиям, а также его функциональность, надежность, производительность и безопасность. Важно отметить, что тестирование не только находит ошибки, но и помогает разработчикам понять, как улучшить качество продукта.

**2. Цели тестирования, разработки и выполнения тестов**

Основные цели тестирования включают:

* **Обнаружение дефектов:** Выявление ошибок, которые могут негативно повлиять на функционирование ПО.
* **Оценка качества:** Подтверждение того, что программное обеспечение соответствует установленным стандартам и требованиям.
* **Улучшение пользовательского опыта:** Обеспечение удобства и удовлетворенности конечных пользователей.
* **Поддержка разработки:** Предоставление информации разработчикам о найденных проблемах для их исправления.
* **Минимизация рисков:** Снижение вероятности возникновения серьезных сбоев и ошибок в работе ПО.

**3. Тестирование QC и QA**

* **Quality Control (QC)** фокусируется на проверке и контроле качества конечного продукта. Это включает в себя тестирование, целью которого является выявление дефектов в ПО.
* **Quality Assurance (QA)** включает в себя более широкий процесс, направленный на создание систем и процессов, обеспечивающих качество на всех этапах разработки. QA ориентировано на предотвращение дефектов и включает в себя как тестирование, так и процессы управления качеством.

**4. Виды тестирования ПО**

Существует множество видов тестирования ПО, включая:

* **Функциональное тестирование:** Проверка выполнения всех функций и требований системы.
* **Нефункциональное тестирование:** Оценка аспектов, таких как производительность, безопасность, удобство использования и др.
* **Регрессионное тестирование:** Проверка того, что изменения в коде не повлияли на существующую функциональность.
* **Тестирование безопасности:** Проверка уязвимостей системы и её защиты от потенциальных атак.
* **Юзабилити-тестирование:** Оценка удобства и интуитивности интерфейса для пользователей.

**5. Классификация видов тестирования по хронологии выполнения**

Тестирование может быть классифицировано по времени выполнения:

* **Предварительное тестирование (pre-release testing):** Тестирование, проводимое перед выпуском продукта.
* **Пострелизное тестирование (post-release testing):** Тестирование, проводимое после выхода продукта на рынок, чтобы выявить проблемы в реальных условиях эксплуатации.

**6. Классификация видов тестирования по формальности**

* **Формальное тестирование:** Процесс, который включает в себя строгую документацию и формальные процедуры тестирования, например, тестовые планы и отчеты.
* **Неформальное тестирование:** Менее структурированный подход, который может включать в себя тестирование без строгой документации.

**7. Классификация видов тестирования по исполнению кода**

* **Динамическое тестирование:** Процесс, при котором выполняется исполняемый код и оценивается его работа.
* **Статическое тестирование:** Анализ кода без его выполнения, включая код-ревью и статический анализ.

**8. Классификация видов тестирования по уровню тестирования**

* **Модульное тестирование:** Тестирование отдельных модулей или компонентов приложения.
* **Интеграционное тестирование:** Проверка взаимодействия между модулями.
* **Системное тестирование:** Оценка всей системы как единого целого.
* **Приемочное тестирование:** Тестирование, проводимое для проверки соответствия требованиям и ожиданиям пользователей.

**9. Классификация видов тестирования по исполнителям тестирования**

* **Тестирование разработчиками:** Тесты, выполняемые самими разработчиками ПО.
* **Тестирование тестировщиками:** Тесты, проводимые отдельными командами тестировщиков.
* **Пользовательское тестирование:** Тестирование, проводимое конечными пользователями, чтобы выявить проблемы в реальных условиях использования.

**10. Классификация видов тестирования по целям**

* **Функциональное тестирование:** Проверка функциональности согласно требованиям.
* **Нефункциональное тестирование:** Оценка нефункциональных аспектов, таких как производительность, безопасность и др.

**11. Классификация видов тестирования по степени автоматизации**

* **Автоматизированное тестирование:** Тестирование, проводимое с помощью автоматизированных инструментов и скриптов.
* **Ручное тестирование:** Тестирование, выполняемое вручную тестировщиками.

**12. Классификация видов тестирования по позитивности сценария**

* **Позитивное тестирование:** Проверка правильного поведения системы при корректных входных данных.
* **Негативное тестирование:** Проверка системы на корректность обработки некорректных входных данных.

**13. Классификация видов тестирования по знанию системы**

* **Черный ящик:** Тестирование без знания внутренней структуры кода.
* **Белый ящик:** Тестирование с знанием внутренней структуры и логики системы.

**14. Классификация видов тестирования по разработке тестовых сценариев**

* **Тестирование по спецификациям:** Тесты разрабатываются на основе требований и спецификаций.
* **Тестирование по сценариям:** Разработка тестов на основе сценариев использования.

**15. Что такое баг**

Баг (от англ. bug) — это ошибка или дефект в программе, приводящий к неправильному поведению системы или её компонентов. Баги могут проявляться в виде сбоев, неправильных расчетов, зависаний или неверного отображения данных. Основной целью тестирования является выявление таких багов и их устранение для повышения качества ПО.

**16. Классификация ошибок**

Ошибки в программном обеспечении можно классифицировать по нескольким критериям. Основные классификации включают:

* **По этапу разработки:** ошибки могут быть выявлены на разных этапах: проектирования, кодирования, тестирования и эксплуатации.
* **По типу:** это могут быть синтаксические ошибки, логические ошибки, ошибки времени выполнения и др.
* **По влиянию на систему:** критические ошибки могут полностью блокировать работу программы, в то время как незначительные могут лишь ухудшать пользовательский опыт.

**17. Синтаксические ошибки в коде**

Синтаксические ошибки возникают, когда код написан с нарушением правил языка программирования. Например, отсутствие точки с запятой в конце строки или неправильное использование скобок. Эти ошибки обычно обнаруживаются на этапе компиляции, и компилятор сообщает об этом разработчику, не позволяя завершить процесс сборки программы. Исправление синтаксических ошибок необходимо для того, чтобы программа могла быть успешно скомпилирована и запущена.

**18. Предупреждения компилятора**

Предупреждения компилятора (warnings) — это сообщения, которые указывают на потенциальные проблемы в коде, которые не являются критическими, но могут привести к ошибкам или неправильному поведению программы. Например, компилятор может предупредить о неинициализированной переменной. Хотя программа может скомпилироваться и запуститься, игнорирование этих предупреждений может привести к ошибкам в дальнейшем. Разработчикам рекомендуется обращать внимание на предупреждения и исправлять их, чтобы повысить качество кода.

**19. Ошибки времени выполнения, смысловые ошибки**

Ошибки времени выполнения (runtime errors) происходят во время выполнения программы, когда код, казалось бы, корректен, но встречает неожиданные условия. Например, деление на ноль или обращение к элементу массива за пределами его границ. Смысловые ошибки (логические ошибки) — это ошибки, которые не вызывают аварийного завершения программы, но приводят к неверным результатам. Например, если вместо сложения двух чисел происходит вычитание, программа выполнится, но результат будет неправильным. Такие ошибки часто сложно выявить, так как они не проявляются на этапе компиляции и могут требовать тщательного тестирования и анализа.

**20. Классификация ошибок по размеру**

Ошибки можно классифицировать по их размеру:

* **Showstoppers:** критические ошибки, которые останавливают тестирование или эксплуатацию программы.
* **Серьезные ошибки:** ошибки, которые влияют на важные функции, но не останавливают работу программы.
* **Незначительные ошибки:** ошибки, которые не влияют на функционирование системы, но могут ухудшать пользовательский опыт или вызывать небольшие неудобства.

**21. Классификация ошибок по времени появления**

Ошибки можно классифицировать по времени их появления:

* **Постоянные ошибки:** ошибки, которые возникают при каждом запуске программы.
* **Иногда возникающие ошибки:** ошибки, которые проявляются лишь в определенных условиях или с определенной частотой.
* **Ошибки, зависящие от среды:** ошибки, которые могут возникать только на машинах с конкретными настройками или программным обеспечением.

**22. Классификация ошибок по месту и направлению**

Ошибки могут классифицироваться в зависимости от их места возникновения и характера:

* **Ошибки пользовательского интерфейса:** ошибки, которые касаются дизайна и взаимодействия с пользователем.
* **Ошибки в системах обработки данных:** ошибки, которые возникают при обработке и интерпретации данных.
* **Ошибки вычислений:** ошибки, возникающие при выполнении арифметических операций.
* **Ошибки контроля потока:** ошибки, связанные с логикой выполнения программы (например, бесконечные циклы).

**23. Разновидности ошибок**

Существуют различные разновидности ошибок, в том числе:

* **Борбаг:** легко обнаруживаемая ошибка, которая стабильно проявляется.
* **Гейзенбаг:** трудноуловимая ошибка, которая исчезает или изменяет свои свойства при попытке её обнаружить.
* **Мандельбаг:** ошибка с хаотичным поведением, трудно предсказуемая.
* **Шрединбаг:** ошибка, которая не проявляется, пока не будет активирована определённым действием.

**24. Поиск и исправление ошибок**

Поиск и исправление ошибок (debugging) — это процесс, в который входят анализ кода, воспроизведение ошибок и использование инструментов отладки. Разработчики используют специальные программы-отладчики, такие как GDB для UNIX-подобных систем или WinDbg для Windows, чтобы выявить и устранить ошибки в программном обеспечении. Процесс может включать в себя изучение логов, использование точек останова и анализ поведения программы.

**25. Серьезность ошибок**

Серьезность ошибок делится на несколько уровней:

* **S1 — Блокирующая:** ошибка, которая полностью останавливает работу приложения.
* **S2 — Критическая:** ошибка, которая нарушает основную бизнес-логику или вызывает серьёзные проблемы, но приложение всё ещё может функционировать.
* **S3 — Значительная:** ошибка, которая вызывает незначительные проблемы и может быть обойдена.
* **S4 — Незначительная:** ошибка, которая не влияет на функционирование системы, например, ошибки в пользовательском интерфейсе.
* **S5 — Тривиальная:** малозначительная ошибка, не оказывающая заметного влияния на пользовательский опыт.

**26. Приоритеты ошибок**

Приоритет ошибок указывает на очередность их устранения:

* **P1 — Высокий:** ошибка должна быть исправлена немедленно, так как её наличие критично для проекта.
* **P2 — Средний:** ошибка требует исправления, но не критична для работы системы.
* **P3 — Низкий:** ошибка может быть исправлена позже, так как её влияние незначительно.

**27. Виды и типы мобильных продуктов**

Существует несколько типов мобильных приложений:

* **Мобильные веб-приложения:** представляют собой адаптированные веб-сайты, которые открываются в мобильных браузерах. Преимущества включают легкость доступа и обновления, но отсутствие автономного режима.
* **Нативные приложения:** разработаны для конкретной платформы (Android, iOS) и используют все возможности устройства. Они имеют высокую производительность и пользовательский интерфейс, но требуют значительных затрат на разработку.
* **Гибридные приложения:** комбинируют характеристики нативных и веб-приложений, что позволяет использовать преимущества обеих технологий. Они более экономичны и легче в распространении, но могут уступать по производительности нативным приложениям.

**28. Особенности тестирования web мобильных продуктов**

Тестирование web мобильных продуктов имеет свои уникальные особенности. Во-первых, мобильные устройства обладают различными размерами экранов и разрешениями, что требует проверки адаптивности интерфейса. Во-вторых, использование разных мобильных браузеров и операционных систем также вносит разнообразие в тестирование. Необходимо учитывать различные условия подключения к интернету, включая 3G, 4G и Wi-Fi, и проверять, как приложение ведет себя в условиях переменной пропускной способности. Кроме того, следует тестировать работу с сенсорными экранами и проверять, как приложение реагирует на различные жесты, такие как прокрутка, сжатие и т. д.

**29. Особенности тестирования нативных мобильных продуктов**

Тестирование нативных мобильных приложений включает в себя несколько аспектов. Поскольку эти приложения разрабатываются для конкретных платформ, важно учитывать их уникальные функции и API. Тестировщики должны проверять взаимодействие приложения с аппаратными возможностями устройства, такими как камера, GPS и сенсоры. Нативные приложения также должны проходить тестирование на производительность, чтобы обеспечить быструю и плавную работу на мобильных устройствах с ограниченными ресурсами. Необходимо также проверить функциональность при различных условиях использования сети, поскольку нативные приложения часто зависят от интернет-соединения для выполнения своих задач.

**30. Особенности тестирования гибридных мобильных продуктов**

Гибридные мобильные приложения, сочетающие в себе элементы веб-приложений и нативных, требуют специфического подхода к тестированию. Во-первых, тестировщики должны оценивать производительность приложения как на веб-стороне (HTML, CSS, JavaScript), так и на нативной (взаимодействие с API платформы). Поскольку гибридные приложения зависят от подключения к интернету, тестирование должно включать различные сценарии с разной скоростью и доступностью сети. Также важно протестировать кроссбраузерную совместимость, поскольку гибридные приложения могут работать на разных устройствах и платформах.

**31. Основные виды тестирования web приложения**

Основные виды тестирования веб-приложений включают:

* **Функциональное тестирование:** проверка выполнения всех функций приложения согласно требованиям.
* **Юзабилити-тестирование:** оценка удобства и интуитивности интерфейса.
* **Тестирование совместимости:** проверка работы приложения на различных браузерах и операционных системах.
* **Тестирование производительности:** оценка времени отклика и нагрузки на систему.
* **Тестирование безопасности:** проверка уязвимостей и защиты данных.

**32. Тестирование функциональности**

Функциональное тестирование веб-приложений включает проверку всех аспектов работы приложения. Это означает, что тестировщики должны проверять ссылки, формы, взаимодействие с базами данных и другие функциональные элементы. Они должны убедиться, что все функции работают корректно, и что приложение отвечает на действия пользователей так, как это предусмотрено в спецификациях. Тестирование включает в себя как позитивные, так и негативные сценарии, чтобы удостовериться, что система корректно обрабатывает как правильные, так и неправильные входные данные.

**33. Тестирование удобства использования**

Юзабилити-тестирование направлено на оценку удобства и интуитивности интерфейса. Оно включает в себя анализ того, как легко пользователи могут ориентироваться на сайте, находить необходимую информацию и выполнять действия. Тестировщики проводят наблюдения за пользователями, чтобы выявить возможные затруднения, а также собирают отзывы о том, что пользователи считают удобным или неудобным. Основные факторы, которые учитываются, — это ясность инструкций, логичность навигации, внешний вид сайта и общая удовлетворенность пользователей.

**34. Тестирование интерфейса**

Тестирование интерфейса фокусируется на проверке графического пользовательского интерфейса (GUI) веб-приложения. Это включает в себя проверку всех элементов интерфейса, таких как кнопки, поля ввода, выпадающие списки и другие компоненты. Тестировщики должны удостовериться, что все элементы отображаются корректно, функционируют согласно требованиям и обеспечивают удобное взаимодействие с пользователем. Также важно проверить, как интерфейс реагирует на различные разрешения экранов и устройства, что особенно актуально для мобильных приложений.

**35. Тестирование совместимости**

Тестирование совместимости направлено на проверку работы веб-приложения на различных браузерах, операционных системах и устройствах. Это важно, поскольку пользователи могут использовать различные комбинации для доступа к приложению. Тестировщики должны проверить, как приложение отображается и работает на популярных браузерах (например, Chrome, Firefox, Safari, Internet Explorer) и на различных устройствах (мобильные телефоны, планшеты, десктопы). Также следует учитывать совместимость с различными версиями браузеров и операционных систем.

**36. Тестирование производительности и скорости загрузки сайта**

Тестирование производительности включает в себя оценку времени отклика приложения и его способности обрабатывать нагрузки. Это включает в себя нагрузочное тестирование, при котором проверяется, как система справляется с большим количеством одновременно работающих пользователей, и стрессовое тестирование, которое направлено на выявление пределов производительности системы. Важно также проверять скорость загрузки страниц, так как задержки могут негативно сказаться на пользовательском опыте.

**37. Тестирование безопасности**

Тестирование безопасности веб-приложений нацелено на выявление уязвимостей, которые могут быть использованы злоумышленниками для доступа к системе или нарушения её работы. Это может включать проверку на наличие SQL-инъекций, XSS (межсайтового скриптинга), а также тестирование механизмов авторизации и шифрования данных. Важной частью тестирования безопасности является использование инструментов для автоматизированного поиска уязвимостей, а также проведение ручных тестов для более глубокого анализа.

**38. Особенности тестирования web приложения**

Тестирование веб-приложений имеет свои особенности, такие как необходимость учитывать различные браузеры и устройства, адаптивность дизайна, а также множество вариантов пользовательского поведения. Важно также тестировать приложения в условиях реальной эксплуатации, что может включать тестирование с использованием различных скоростей интернет-соединения. Тестировщики должны также обращать внимание на производительность, безопасность и удобство использования приложения, чтобы обеспечить его успешную работу в реальных условиях.