**1. Тестирование ПО. Формальное определение, основные понятия**

Тестирование программного обеспечения (ПО) — это процесс проверки и оценки качества ПО, направленный на выявление ошибок, дефектов и недостатков в его работе. Цель тестирования ПО — обеспечение качества продукта и повышение уверенности в его работе.

Основные понятия в тестировании ПО включают в себя:

1. Тест-кейс: описание последовательности действий, которые должен выполнить тестировщик для проверки определенной функциональности ПО.
2. Тест-план: документ, содержащий информацию о том, какие тесты будут проведены, какие ресурсы для этого необходимы, какие результаты ожидаются и как будет оцениваться успех тестирования.
3. Тест-сьют: набор тестов, объединенных по определенному критерию (например, функциональности, приоритету, типу тестирования).
4. Баг (дефект): ошибки в работе ПО, которые должны быть исправлены разработчиками.
5. Регрессионное тестирование: повторное тестирование функциональности ПО после внесения изменений, чтобы проверить, не появились ли новые дефекты.
6. Тестирование производительности: проверка работы ПО при нагрузке для определения его скорости, масштабируемости и реакции на различные нагрузки.
7. Автоматизированное тестирование: использование специальных инструментов и программ для автоматического выполнения тестовых сценариев и анализа результатов тестирования.

**2. Организация тестирования ПО**

Организация тестирования программного обеспечения является важным этапом в разработке ПО, который включает в себя следующие этапы:

1. Планирование тестирования: на этом этапе определяются цели тестирования, составляется тест-план, выбираются тест-кейсы и оценивается необходимый объем ресурсов для проведения тестирования.
2. Создание тестовых случаев: разработка тест-кейсов и тест-сьютов на основе спецификации требований.
3. Подготовка тестового окружения: подготовка тестовых сред, которые могут быть как аппаратными, так и программными, для выполнения тестовых случаев.
4. Выполнение тестовых случаев: проведение тестовых сценариев в соответствии с тест-планом и запись результатов тестирования.
5. Оценка результатов тестирования: анализ результатов тестирования, выявление дефектов и оценка качества ПО.
6. Исправление ошибок и повторное тестирование: исправление дефектов, которые были выявлены в результате тестирования, и проведение повторного тестирования для подтверждения исправления.
7. Отчетность: составление отчетов о результатах тестирования и передача их соответствующим заинтересованным сторонам.

Важно учитывать, что организация тестирования ПО должна осуществляться в тесном сотрудничестве с другими отделами и специалистами, такими как разработчики, аналитики, менеджеры проекта и пользователи, чтобы обеспечить качество ПО и удовлетворение потребностей пользователей.

**3. Основные подходы к поиску ошибок. Проблемы при тестировании**

Основными подходами к поиску ошибок в программном обеспечении являются:

1. Тестирование черного ящика (black-box testing): при этом подходе тестируются функциональные возможности ПО без знания внутренней реализации. Тестировщик сосредотачивается на входных и выходных данных, а также на правильности выполнения действий ПО в соответствии с требованиями.
2. Тестирование белого ящика (white-box testing): этот подход к тестированию основывается на знании внутренней структуры ПО и тестирует отдельные части программного кода, а также проверяет взаимодействие между ними.
3. Тестирование серого ящика (gray-box testing): при таком подходе тестируются функциональные возможности ПО с некоторыми знаниями о внутренней структуре и реализации.

Проблемы при тестировании могут возникнуть из-за различных факторов, например:

1. Недостаточное покрытие тестами: это может привести к неполному обнаружению дефектов в ПО.
2. Ошибки в тестовых данных: если тестовые данные содержат ошибки или недостаточно представительны, это может привести к неправильному оцениванию качества ПО.
3. Недостаточное понимание требований: если тестировщик не полностью понимает требования, то тестирование может быть неполным или некорректным.
4. Недостаточная экспертиза: если тестирование проводится некомпетентными специалистами, это может привести к неправильной оценке качества ПО и пропуску дефектов.
5. Недостаточное внимание к деталям: если тестировщик не обращает внимание на детали, то это может привести к неполному обнаружению дефектов в ПО.
6. Неправильное планирование тестирования: если планирование тестирования проводится неправильно, то это может привести к неполному тестированию или недостаточному выделению ресурсов на тестирование.

**4. Классы критериев тестирования**

Классы критериев тестирования - это различные подходы к оценке качества программного обеспечения. Различные классы критериев могут использоваться для различных типов тестирования и в различных контекстах.

Ниже приведены основные классы критериев тестирования:

1. Критерии функциональности: используются для проверки соответствия функциональных требований к ПО.
2. Критерии надежности: используются для проверки надежности и стабильности ПО при различных условиях.
3. Критерии производительности: используются для оценки производительности ПО в различных сценариях использования.
4. Критерии совместимости: используются для проверки совместимости ПО с различными конфигурациями аппаратного и программного обеспечения.
5. Критерии удобства использования: используются для оценки удобства использования ПО, включая его эргономику, удобство интерфейса и т.д.
6. Критерии безопасности: используются для проверки защиты ПО от внешних угроз и обеспечения безопасности данных.
7. Критерии сопровождаемости: используются для проверки легкости сопровождения и обслуживания ПО.
8. Критерии совместной работы: используются для проверки совместной работы ПО с другими системами и программами.

Каждый из этих классов критериев имеет свои специфические методы и инструменты для оценки качества программного обеспечения. Использование этих критериев может помочь улучшить качество ПО и обеспечить его соответствие требованиям.

**5. Идеальный критерия тестирования**

В теории тестирования программного обеспечения не существует идеального критерия тестирования, который бы позволял полностью охватить все возможные ошибки в программе. Это связано с тем, что программное обеспечение может иметь сложную логику работы, различные варианты использования и взаимодействовать с различным окружением.

**6. Структурные критерии тестирования**

Структурные критерии тестирования используются для оценки тестового покрытия программного кода. Они основаны на анализе структуры программы, включая ее составляющие элементы, такие как операторы, условные операторы, циклы и подпрограммы.

Существует несколько типов структурных критериев тестирования, включая критерии покрытия утверждений, критерии покрытия решений и критерии покрытия путей. Каждый критерий описывает определенный аспект структуры программы, который должен быть протестирован.

Критерий покрытия утверждений проверяет каждое логическое утверждение в программе на истинность и ложность, чтобы гарантировать, что все пути выполнения кода были испытаны.

Критерий покрытия решений требует, чтобы каждое условие в программе было испытано как минимум один раз и что каждая возможная комбинация условий была проверена.

Критерий покрытия путей требует, чтобы каждый возможный путь выполнения программы был испытан как минимум один раз.

Структурные критерии тестирования могут помочь обеспечить более полное покрытие программного кода и выявить потенциальные ошибки, которые могут быть упущены другими методами тестирования. Однако они не являются единственным способом тестирования программного обеспечения и могут быть дополнены другими методами, такими как функциональное тестирование и тестирование производительности.

**7. Функциональные критерии тестирования**

Функциональные критерии тестирования используются для оценки соответствия программного обеспечения его функциональным требованиям. Они фокусируются на тестировании функциональности программы, то есть на проверке ее способности выполнять заявленные функции и действия.

Функциональные критерии тестирования могут включать в себя проверку входных и выходных данных программы, проверку ее реакции на определенные входные данные и проверку ее способности выполнения определенных действий.

Критерии тестирования функциональности могут варьироваться в зависимости от требований и характеристик тестируемой системы. Например, для веб-приложения критериями могут быть проверка корректной работы ссылок и форм, проверка отображения данных на страницах и проверка функций авторизации и регистрации пользователей.

Функциональное тестирование важно для обеспечения качества программного обеспечения и удовлетворения потребностей пользователей. Оно может помочь выявить ошибки, связанные с функциональностью программы, такие как неверные результаты расчетов, неправильное отображение данных и некорректное поведение программы при определенных условиях.

В целом, функциональные критерии тестирования являются важной составляющей процесса тестирования программного обеспечения, помогающей обеспечить соответствие программы требованиям и ожиданиям пользователей.

**8. Стохастические критерии тестирования**

Стохастические критерии тестирования основаны на вероятностном анализе тестовых данных. Они используются для оценки вероятности обнаружения ошибок в программном обеспечении при определенных условиях.

Стохастические критерии тестирования могут включать в себя генерацию случайных тестовых данных или выбор определенных тестовых данных на основе статистического анализа. Также могут использоваться методы Монте-Карло для симуляции случайных событий и оценки вероятности их возникновения.

Применение стохастических критериев тестирования может помочь выявить проблемы, которые могут не быть обнаружены другими методами тестирования. Однако они могут быть ограничены в своей эффективности, так как вероятность обнаружения ошибок может быть низкой, особенно при использовании небольшого количества тестовых данных.

В целом, стохастические критерии тестирования могут быть полезными для дополнения других методов тестирования, таких как функциональное тестирование и тестирование производительности. Они могут помочь выявить проблемы, которые могут быть упущены другими методами тестирования, и обеспечить более полное покрытие программного кода.

**9. Мутационный критерий тестирования**

Мутационный критерий тестирования является методом оценки эффективности набора тестов на основе создания искусственных ошибок в программном коде. Он основан на предположении, что хороший набор тестов должен обнаруживать большинство ошибок в программном коде.

Процесс мутационного тестирования включает в себя создание мутантов - искусственных вариантов программного кода, которые модифицируются с целью создания ошибок. Затем для каждого мутанта выполняются тесты из набора тестов, и если тесты успешно проходят, то мутант убивается, что означает, что он был эффективно обнаружен. Если же тесты не проходят, то мутант считается выжившим, и это указывает на недостаточную эффективность набора тестов.

Мутационный критерий тестирования позволяет оценить качество набора тестов на основе количества обнаруженных мутантов. Чем больше мутантов было обнаружено, тем эффективнее набор тестов.

Преимуществом мутационного тестирования является его способность обнаружить сложные ошибки в программном коде, которые могут быть упущены другими методами тестирования. Однако этот метод также может быть затратным и трудоемким, так как требуется создание искусственных мутантов и запуск тестов для каждого из них.

В целом, мутационный критерий тестирования может быть полезным инструментом для оценки качества набора тестов и обеспечения более высокого уровня качества программного обеспечения.

**10. Модульное тестирование ПО**

Модульное тестирование ПО - это метод тестирования, при котором отдельные модули программного кода тестируются отдельно от других модулей и от остальной части программы в целом. Цель модульного тестирования состоит в проверке правильности работы каждого модуля в изоляции от остальной программы.

Модульное тестирование может быть выполнено как вручную, так и автоматизированно с использованием специальных инструментов. Обычно для модульного тестирования используются фреймворки и библиотеки, которые позволяют разработчикам легко создавать и запускать тесты на отдельные модули.

**11. Разработка ПО через тестирование**

Разработка ПО через тестирование (Test Driven Development, TDD) – это методология разработки программного обеспечения, которая предполагает создание тестов перед написанием кода. В TDD разработка начинается с написания тестов для каждой новой функциональности, затем написание кода для прохождения этих тестов и, наконец, рефакторинг кода для улучшения его качества.

Такой подход помогает улучшить качество кода и уменьшить количество ошибок в нем. Он также способствует улучшению понимания требований к программному обеспечению и облегчает процесс тестирования.

Однако, следует отметить, что TDD не является универсальным методом разработки и может быть неэффективен в некоторых ситуациях. Также его использование требует дополнительных усилий и времени на написание тестов, что может замедлить процесс разработки.

В целом, TDD может быть эффективным методом разработки программного обеспечения, особенно при работе с большими проектами и сложными требованиями.

**12. Интеграционное тестирование ПО**

Интеграционное тестирование ПО - это процесс тестирования, который направлен на проверку взаимодействия между различными компонентами системы, которые взаимодействуют друг с другом. Целью интеграционного тестирования является выявление ошибок, которые могут возникнуть в результате интеграции компонентов.

Интеграционное тестирование ПО выполняется в несколько этапов. На первом этапе тестируются отдельные компоненты ПО, затем производится их интеграция, и уже после этого проводятся интеграционные тесты. Тесты могут проводиться на различных уровнях интеграции: модульном, подсистемном, системном и т.д.

Важным аспектом интеграционного тестирования является наличие надежной тестовой среды, которая позволяет создавать условия для тестирования взаимодействия компонентов. Тестовая среда может включать в себя различные инструменты и фреймворки для автоматизации тестирования, которые помогают ускорить и упростить процесс тестирования.

Интеграционное тестирование является важным этапом в цикле разработки ПО, так как позволяет выявить ошибки, которые могут привести к сбоям в работе системы. При проведении интеграционного тестирования следует уделять внимание как функциональным, так и нефункциональным требованиям, таким как производительность, масштабируемость, надежность и безопасность системы.

**13. Тестирование интерфейсов**

Тестирование интерфейсов (UI-тестирование) является одним из важных этапов тестирования программного обеспечения, который направлен на проверку правильности и соответствия пользовательского интерфейса заданным требованиям.

UI-тестирование проводится для проверки всех элементов интерфейса, включая поля ввода, кнопки, меню, диалоговые окна, таблицы и т.д. В процессе тестирования проверяются такие аспекты, как доступность, удобство использования, визуальная привлекательность и т.д.

Для проведения UI-тестирования используются различные инструменты, включая автоматические тестовые сценарии, ручное тестирование, а также средства записи и воспроизведения тестовых сценариев. Важно отметить, что при тестировании интерфейсов необходимо учитывать различные аспекты, такие как различные языки, операционные системы, разрешения экрана и т.д.

Основной задачей тестирования интерфейсов является проверка на соответствие требованиям и ожиданиям конечного пользователя. На основе результатов тестирования могут быть выявлены проблемы с интерфейсом, которые затрудняют использование программного продукта конечным пользователям.

В целом, тестирование интерфейсов является важной частью тестирования программного обеспечения, которая позволяет убедиться в правильности работы пользовательского интерфейса и улучшить пользовательский опыт.

**14. Тестирование с нагрузкой**

Тестирование с нагрузкой (Load testing) является одним из видов функционального тестирования программного обеспечения, которое позволяет проверить работоспособность приложения при различных нагрузках и выявить возможные проблемы, связанные с производительностью.

В процессе тестирования с нагрузкой генерируется большое количество запросов к приложению, которые имитируют действия реальных пользователей. Это позволяет определить, как быстро приложение отвечает на запросы и какие ресурсы использует.

Для проведения тестирования с нагрузкой используются специальные инструменты, которые могут генерировать запросы с различными параметрами, такими как количество одновременных пользователей, объем передаваемых данных и т.д. Результаты тестирования позволяют оценить производительность приложения и определить максимальное количество пользователей, которые могут работать с приложением одновременно.

Основная цель тестирования с нагрузкой - определить пределы производительности приложения и выявить проблемы, которые могут возникнуть при большой нагрузке. Например, это может быть связано с длительным временем ответа приложения или недостаточным объемом выделенной памяти.

В целом, тестирование с нагрузкой является важным этапом тестирования, которое позволяет выявить проблемы производительности и улучшить качество программного обеспечения, что является важным для обеспечения эффективной работы приложений в реальных условиях использования.

**15. Системное тестирование**

Системное тестирование (System testing) - это процесс тестирования всей системы в целом, а не отдельных ее компонентов или модулей. Целью системного тестирования является проверка соответствия системы ее требованиям и спецификациям, а также выявление возможных ошибок, дефектов и проблем, которые могут возникнуть при работе системы.

В процессе системного тестирования проверяются различные аспекты системы, такие как функциональность, производительность, надежность, безопасность и т.д. Для этого используются различные методы и инструменты, такие как тест-кейсы, тест-сьюты, автоматизированные сценарии и т.д.

Одной из особенностей системного тестирования является то, что оно проводится в условиях, максимально приближенных к реальным. Тестирование проводится на фактической аппаратуре и программном обеспечении, которые будут использоваться в работе системы.

Кроме того, системное тестирование может включать в себя такие виды тестирования, как интеграционное тестирование и приемочное тестирование. При этом системное тестирование выполняется после проведения всех предыдущих этапов тестирования, таких как модульное тестирование и интеграционное тестирование.

В результате проведения системного тестирования можно получить оценку работы системы в целом, выявить проблемы, которые могут возникнуть при использовании системы в реальных условиях, а также принять меры по устранению этих проблем.

**16. Регрессионное тестирование ПО**

Регрессионное тестирование (Regression testing) - это процесс повторного тестирования программного продукта с целью проверки, что изменения, внесенные в программное обеспечение, не повлияли на работу уже существующей функциональности и не вызвали появления новых дефектов.

Регрессионное тестирование проводится после внесения каких-либо изменений в код, например, после исправления дефектов, добавления новой функциональности или изменения конфигурации системы. Целью регрессионного тестирования является проверка, что после внесения изменений программное обеспечение продолжает работать стабильно и не произошло никаких непредвиденных изменений в уже существующей функциональности.

В процессе регрессионного тестирования обычно используются заранее разработанные тест-кейсы, которые позволяют проверить основные функциональные возможности программного продукта. Для автоматизации регрессионного тестирования можно использовать специальные инструменты, такие как фреймворки для тестирования, системы управления тестированием и т.д.

Регрессионное тестирование является важной частью процесса разработки программного обеспечения, так как оно позволяет выявить проблемы, которые могут возникнуть после внесения изменений, и принять меры по их устранению. Это также помогает убедиться в том, что программное обеспечение продолжает работать стабильно и соответствует ожиданиям заказчика.