Вопросы

1. Цели и задачи тестирования, что входит в задачи тестировщика

Цели и задачи тестирования:

1. Предоставить информацию заказчику, разработчику, руководителю проекта

2. Информация: обнаруженные несоответствия и рекомендации по их исправлению, рекомендации к усовершенствованию

3. С помощью: средств управления тестированием, отчётов, встреч

Задачи тестировщика:

• Определение миссии тестирования

• Проверка подхода к тестированию

• Проверка стабильности выпуска

• Тестирование и оценка

• Достижение приемлемого результата миссии

• Совершенствование методов и средств тестирования

1. Тестирование пользовательского интерфейса, ручное тестирование интерфейса, плюсы и минусы

В тестирование пользовательского интерфейса входит проверка взаимного расположения элементов, их внешний вид и интуитивность расположения. Кроме того, учитывается переход по элементам при нажатии клавиши Tab и прочее. Ручное тестирование не требует предварительных приготовлений кроме как тестировщика, но некоторые функции могут быть упущены из-за его невнимательности. Кроме того, ручное тестирование не может покрыть все возможные варианты взаимодействия с интерфейсом, однако лучше проверяет лаконичность и удобство интерфейса.

1. Процесс выполнения тестов

Процесс выполнения тестов:

• Выполнение тестов

• Оценка выполнения тестов

• Восстановление после сбойных тестов

• Проверка результатов

• Исследование неожиданных результатов

• Запись ошибок

1. Тестирование пользовательского интерфейса, автоматическое тестирование интерфейса, плюсы и минусы

В тестирование пользовательского интерфейса входит проверка взаимного расположения элементов, их внешний вид и интуитивность расположения. Кроме того, учитывается переход по элементам при нажатии клавиши Tab и прочее. Автоматическое тестирование требует создания тестов и тестовой среды, но если тест написан правильно, то по эффективности он будет выше чем ручной, ибо более защищен от невнимательности чем ручной.

1. Участники процесса тестирования, их назначение

Участники процесса тестирования:

1. Менеджер проекта – обеспечение ресурсами, координация работ

2. Разработчик, технический писатель – исправление найденных ошибок

3. Архитектор – обеспечение целостности проекта в процессе исправления ошибок

4. Интегратор (Релиз менеджер) – контроль и выпуск версий ПО

5. Аналитик – установка приоритетов, связанных с необходимостью и сложностью исправления найденных дефектов

6. Тестировщик – несет ответственность за процесс тестирования в целом

1. Жизненный цикл ПО. Этапы.

Жизненный цикл ПО начинается с идеи о задумке какого – либо ПО. Конец – когда ПО выводится из эксплуатации. Стадии:

1. Анализ требований

2. Проектирование

3. Разработка/кодирование

4. Тестирование

5. Ввод в эксплуатацию и доработка

1. Виды тестирования, интеграционное тестирование, достоинства и недостатки.

Интеграционное тестирование:

• Проверка связи между компонентами, а также взаимодействия с различными частями системы (ОС, оборудование и т.д.)

• Поиск дефектов, связанных с ошибками в реализации и интерпретации взаимодействия между модулями

• Требует создания тестового окружения

• Метод «Белого ящика»

1. Виды тестирования, модульное тестирование, достоинства и недостатки.

Модульное тестирование:

• Проверка отдельно взятых модулей, функций или классов

• Выявление локализованных в модуле ошибок в алгоритмах или реализации алгоритмов

• Требует создания тестового окружения

• Метод «Белого ящика»

1. Виды тестирования, системное тестирование, достоинства и недостатки.

Системное тестирование:

• Проверка соответствия установленным требованиям

• Выявление дефектов, связанных с работой системы в целом:

• Отсутствующая функциональность

• Неверное использование ресурсов системы

• Непредусмотренные комбинации данных пользовательского уровня

• Несовместимость с окружением

• Непредусмотренные сценарии использования

• Неудобство в применении и т.п.

• Метод «Черного ящика»

1. Функциональное тестирование, дать описание.

Функциональное тестирование – рассматривает заранее указанное поведение и основывается на анализе спецификаций функциональности компонента или системы в целом. Может приводится в аспекте требований и бизнес процессов. Может приводится на всех уровнях тестирования.

1. Инсталляционное тестирование, на что оно направленно.

• Инсталляционное – проверка успешной установки и настройки, удаления обновления ПО при различном программном и аппаратном окружении, призванное также оценить работоспособность системы после завершения работы инсталлятора.

1. Тестирование безопасности на что оно направленно.

• Безопасности – используется для проверки безопасности системы, а также для анализа рисков, связанных с обеспечением целостного подхода к защите приложения, атак хакеров, вирусов, несанкционированного доступа к конфиденциальным данным

1. Тестирование удобства использования, на что оно направленно.

• Удобства использования – направлено на установление степени удобства использования, обучаемости, понятности и привлекательности для пользователей разрабатываемого продукта в контексте заданных условий.

1. Что такое требование, источники требований, методы выявления требований.

Требования – условие или возможность требуемая пользователем для решения задач или достижения целей

Требования – условие или возможность которые должны удовлетворяться системой или компонентом системы, или которыми система или компонент должна обладать для обеспечения условий контракта, стандартов, спецификаций или др. регулирующими документами

Требования – документальная репрезентация условий или возможностей, перечисленных в предыдущих пунктах

1. Тестовая стратегия, что включает в себя тестовая стратегия.

Тестовая стратегия – общий и не сильно детализированный план контроля качество ПО, охватывающий длительный период времени, главная цель которого достижение и обеспечение высокого качества ПО.

Тестовые стратегии бывают: конкретными, практичными и основательными.

Стратегия включает:

• Типы тестов, которые нужно выполнять для данного функционала системы

• Описание необходимых подходов с точки зрения целей тестирования

• Описания или требования к необходимым для проведения тестирования инструментам и инфраструктуре

Стратегия отвечает на вопросы:

• Каким образом тестирование даст ответ что данный функционал работает

• Что нужно сделать и чем пользоваться из инструментальных средств для достижения целей

• Когда определенный функционал будет тестироваться и соответственно, когда ожидать результат

• Как минимизировать затраты на тестирование

1. План тестирования, цели плана тестирования.

План тестирования – это документ, в котором определены объем, ресурсы, а также описан календарный план работ по тестированию. В нем определяются выполняемые тесты, тестируемые элементы, задачи тестирования, сотрудники, ответственные за выполнение каждой из задач, а также указываются вероятности возникновения непредвиденных обстоятельств и описывается, какие меры нужно при этом принимать. За план тестирования ответственен тест-менеджер или руководитель группы тестирования.

Цели плана тестирования:

• Понять, что, как, когда, кем будет кем не будет проверяться

• Донести инфу до заказчика и команды

1. План тестирования, критерии начала и окончания тестирования.

План тестирования – это документ, в котором определены объем, ресурсы, а также описан календарный план работ по тестированию. В нем определяются выполняемые тесты, тестируемые элементы, задачи тестирования, сотрудники, ответственные за выполнение каждой из задач, а также указываются вероятности возникновения непредвиденных обстоятельств и описывается, какие меры нужно при этом принимать. За план тестирования ответственен тест-менеджер или руководитель группы тестирования.

Критерии начала окончания тестирования

Начала:

• Готовность функционала или его части

• Готовность тестовой среды

• Предоставлена актуальная документация

• Предоставлены ресурсы

Окончания:

• Выполнены все тесты

• Все тесты успешны

• Зарегистрированы все обнаруженные дефекты, нет открытых дефектов приоритета Критичный или Высокий

• Производительность не упала

• Безопасность в пределах допустимого

1. Тест-дизайн, цели и задачи тест-дизайна.

Тест – дизайн – это этап процесса тестирования ПО, на котором проектируются и создаются тестовые случаи в соответствии с определенными ранее критериями качества и целями тестирования. Роли ответственные за тест-дизайн: тест-аналитик: что тестировать и тест-дизайнер: как тестировать.

Цели тест-дизайна:

• Придумать тесты, которые обнаружат наиболее серьезные ошибки продукта

• Минимизировать кол-во тестов, необходимых для нахождения большинства серьезных ошибок

Задачи тест-дизайна

• Анализ имеющихся проектных артефактов: документация, модели, исполняемый код и т.д.

• Составление списка функций продукта

• Построение таблиц принятия решений

• Написание спецификации по тест-дизайну

• Написание тест-кейсов

• Создание приемочных проверок

• Анализ жалоб пользователей

• Анализ рисков

• Описание процесса тестирования

• Расстановка приоритетов тестирования

1. Тест-кейс, атрибуты тест-кейса, плюсы и минусы тест-кейса.

Тест-кейс – тестовый случай – это артефакт тест-дизайна, описывающий совокупность конкретных условий и параметров, шагов, используемых данных для проверки. Бывает позитивным или негативным в зависимости от результата.

Тест – кейс может быть с положительным результатом, с отрицательным, или с невозможным выполнением.

Атрибуты тест-кейса:

• ID

• Название

• Предусловия

• Шаги

• Ожидаемый результат

Необязательные атрибуты:

• Статус

• Ссылка на требование

• История изменений

• Ссылка на дефекты

• Ссылка на протокол выполнения

• Постусловия

В тест кейсе не должно быть зависимостей от других тест-кейсов, нечеткой формулировки, излишней детализации, отсутствия необходимой для прохождения тест-кейса информации.

Плюсы и минусы тест-кейсов:

+ Можно выполнять новичку

+ Являются базой знаний

- Очень много повторения

- Сложно поддерживать

1. Приоритет и Важность, градации важности дефекта.

• Градации важности дефекта:

• 1 – Блокирующая

• 2 – Критическая

• 3 – Значительная

• 4 – Незначительная

• 5 – Тривиальная

• Градации приоритета дефекта:

• 1 – Высокий

• 2 – Средний

• 3 – Низкий

1. Модульное тестирование, цели модульного тестирования, драйвер, заглушка.

Модульное тестирование:

• Проверка отдельно взятых модулей, функций или классов

• Выявление локализованных в модуле ошибок в алгоритмах или реализации алгоритмов

• Требует создания тестового окружения

• Метод «Белого ящика»

Заглушки и драйвера – вспомогательные части кода или программы, помогающие при выполнении тестов в случае, к примеру, инкрементального тестирования

1. Системное тестирование, объекты выявления при системном тестировании.

Системное тестирование:

• Проверка соответствия установленным требованиям

• Выявление дефектов, связанных с работой системы в целом:

• Отсутствующая функциональность

• Неверное использование ресурсов системы

• Непредусмотренные комбинации данных пользовательского уровня

• Несовместимость с окружением

• Непредусмотренные сценарии использования

• Неудобство в применении и т.п.

• Метод «Черного ящика»

1. Интеграционное тестирование, возможное ошибки при интеграции, объекты интеграционного тестирования

Интеграционное тестирование:

• Проверка связи между компонентами, а также взаимодействия с различными частями системы (ОС, оборудование и т.д.)

• Поиск дефектов, связанных с ошибками в реализации и интерпретации взаимодействия между модулями

• Требует создания тестового окружения

• Метод «Белого ящика»

Возможные ошибки: конфликты работы кода из-за особенностей ОС, ошибка работы с различными частями ПК

1. Что такое тестирование, что такое тест, тестовый случай.

Тестирование – это процесс, позволяющий убедится в том, что в программе нет ошибок, при этом, невозможно отыскать абсолютно все ошибки в программном продукте. Из этого выходит, что тестирование – процесс, позволяющий убедится в том, что программа выполняет свое назначение.

Тестирование – наблюдение за функционированием ПО в специфических условиях с целью определения степени соответствия ПО к требованиям.

Тестирование – одна из дисциплин RUP (Методология разработки ПО).

Тест – специальная искусственно созданная ситуация, выбранная определенным образом, и описание того, какие наблюдения за работой программы нужно сделать для проверки её соответствия некоторому требованию.

Тестовый случай – артефакт либо документ, описывающий совокупность шагов, конкретных условий и параметров, необходимых для проверки реализации тестируемой функции или её части.

Тестовый случай используется для ручного или автоматического тестирования.

1. Тестовое покрытие, что входит в состав тестового набора, критерии покрытия кода

Тестовое покрытие - это мера, которая определяет, насколько хорошо тесты покрывают функциональность приложения. В состав тестового набора входят различные типы тестов, такие как:

1. Unit-тесты – тестирование отдельных модулей (классов, методов) приложения;

2. Интеграционные тесты – тестирование взаимодействия между модулями;

3. Функциональные тесты – проверка функциональности приложения в соответствии с требованиями;

4. Нагрузочные тесты – проверка работоспособности приложения при большой нагрузке;

5. Безопасностные тесты – проверка безопасности приложения.

Критерии покрытия кода включают в себя:

1. Покрытие строк кода – процент строк кода, которые были выполнены во время тестирования;

2. Покрытие ветвей кода – процент выполненных условных операторов (if/else);

3. Покрытие функций – процент выполненных функций;

4. Покрытие классов – процент выполненных классов;

5. Покрытие методов – процент выполненных методов.

1. Ошибка, дефект, отказ. Что это такое, в чем отличия.

Ошибка - это человеческий фактор, который приводит к неправильному выполнению функциональности приложения. Ошибка может возникнуть на любом этапе разработки приложения и может быть исправлена до того, как приложение будет выпущено.

Дефект - это неполадка в приложении, которая может быть вызвана ошибкой. Дефект может возникнуть в любой части приложения и может быть обнаружен во время тестирования или после выпуска приложения. Дефект должен быть исправлен, чтобы обеспечить правильную работу приложения.

Отказ - это ситуация, когда приложение перестает работать правильно. Отказ может быть вызван дефектом или другими проблемами, такими как недостаточная производительность или неправильная конфигурация. Отказ должен быть исправлен как можно скорее, чтобы вернуть приложение к работоспособному состоянию.

1. Тестирование пользовательского интерфейса, задачи тестирования пользовательского интерфейса.

В тестирование пользовательского интерфейса входит проверка взаимного расположения элементов, их внешний вид и интуитивность расположения. Кроме того, учитывается переход по элементам при нажатии клавиши Tab и прочее.

1. Проверка на соответствие дизайну и стилю: проверка, что элементы интерфейса соответствуют дизайну, цветам, шрифтам и другим стилистическим параметрам.

2. Проверка на доступность: проверка, что пользователи с ограниченными возможностями (например, слабовидящие, глухие или люди с ограниченной моторикой) могут использовать приложение без проблем.

3. Проверка на удобство использования: проверка, что пользователи могут легко находить нужные функции и выполнять задачи без необходимости читать инструкции.

4. Проверка на совместимость: проверка, что приложение работает корректно на различных устройствах и платформах.

5. Проверка на безопасность: проверка, что приложение не представляет угрозы для безопасности пользователей и их данных.

6. Проверка на функциональность: проверка, что все функции приложения работают корректно и выполняются в соответствии с требованиями.

7. Проверка на производительность: проверка, что приложение работает быстро и не тормозит при выполнении задач.

8. Проверка на локализацию: проверка, что приложение работает корректно на различных языках и культурах.

1. Верификация и валидация, что это такое, в чем отличия

Верификация – процесс оценки системы или её компонентов с целью определения удовлетворяют ли результаты текущего этапа разработки условиям, сформированным в начале этого этапа. Т.е. выполняются ли наши цели, сроки, задачи по разработке проекта, определенные в начале текущей фазы.

Валидация – определение соответствия разрабатываемого ПО ожиданиям и потребностям пользователя, требованиям к системе.

1. Тестирование пользовательского интерфейса, виды тестирования пользовательского интерфейса.

В тестирование пользовательского интерфейса входит проверка взаимного расположения элементов, их внешний вид и интуитивность расположения. Кроме того, учитывается переход по элементам при нажатии клавиши Tab и прочее.

1. Тестирование функциональности элементов интерфейса: проверка, что все кнопки, поля ввода, выпадающие списки и другие элементы интерфейса работают корректно и выполняют свои функции.

2. Тестирование навигации: проверка, что пользователи могут легко перемещаться по приложению и находить нужную информацию.

3. Тестирование визуального дизайна: проверка, что дизайн интерфейса соответствует требованиям заказчика и создает приятное визуальное впечатление.

4. Тестирование реакции на различные события: проверка, что приложение корректно реагирует на различные действия пользователей, такие как клики на кнопки или изменение данных в полях ввода.

5. Тестирование на соответствие стандартам: проверка, что приложение соответствует стандартам разработки пользовательских интерфейсов, таким как стандарты доступности или принципы юзабилити.

6. Тестирование на мобильных устройствах: проверка, что приложение работает корректно на различных мобильных устройствах и адаптируется к разным размерам экранов.

7. Тестирование на разных браузерах: проверка, что приложение работает корректно на разных браузерах и операционных системах, таких как Chrome, Firefox, Safari или Internet Explorer.

1. Классы эквивалентности. Анализ граничных значений

Класс эквивалентности – это набор значений переменной, который считается эквивалентным.

Анализ граничных значений

Тесты строятся с ориентацией на использование тех величин, которые определяют предельные характеристики тестируемое системы.

1. Выделить классы эквивалентности

2. Определить граничные значения этих классов

3. Понять, к какому классу будет относиться каждая граница

4. Для каждой границы нам нужно провести тесты по проверке значения до границы, на границе, и сразу после границы.

Кол-во тестов = кол-во границ \* 3

1. Матрица принятия решений.

Матрица принятия решений

Описывает логику приложения основываясь на сущностях состояния системы. Каждая decision table должна описывать 1 состояние системы.

Матрица принятия решений — это инструмент для оценки и выбора наилучшего варианта из нескольких возможных. Он особенно полезен, если у вас больше одного варианта решения проблемы и несколько факторов, которые нужно принять во внимание для принятия окончательного решения.