Введение в тестирование ПО.Методы и виды тестирования ПО.

Тестирование - проверка соответствия между реальным и ожидаемым поведением программы, осуществляемая на конечном наборе тестов , выбранном определенным образом.

QA-Обеспчение качества - отвечает за весь процесс разработки поэтому должно быть интегрировано во все этапы разработки : от описания проекта до тестирования , релиза и даже пост-релизного обслуживания.

QC-Контроль качества-Задача QC гарантировать соответствие требованиям(поиск ошибок и их устранение).

QC ориентирован на проверку продукта , включает в себя многие процессы , такие как анализ кода ,технические обзоры , анализ дизайна , тестирование и пр.

Дефект/Баг/Дефект Репорт- это документ , описывающий ситуацию или последовательность действий , которая привела к некорректной работе объекта тестирования , с указанием причин, фактического и ожидаемого результата.

Тестовый сценарий (Test Case) - это артефакт , описывающий совокупность шагов , конкретных условий и параметров , необходимых для проверки реализации тестируемой функции или ее части.

Тестовые данные-набор данных , необходимых для выполнения тестового сценария.

Ими могут быть : логин и пароль или файл для загрузки в программу , на котором загрузчик перестает работать.

Тест дизайн - это этап процесса тестирования ПО , на котором проектируются и создаются тестовые случаи (тест кейсы) , в соответствии с определнным ранее критериями качества и целями тестирования.

В идеале за процесс отвечают : тест-аналитки и тест-дизайнеры , но в реалиях небольших команд это могут делать и рядовые QA специальсты.

1. первым этапом проводится анализ имеющихся проектных артефактов :документация (спецификация , требования , планы), модели , исполняемый код и тд.
2. Написание спецификаии по тест дизайну(Test Design Specification)
3. Проектирование и создание тестовых случаев (Test Cases).

Цели тестирования.

Цель тестирования-минимализация количества существующих дефектов в конечном продукте.*Грамотно организованное тестирование дает гарантии того , что:*

1. система удолитворяет требованиям.
2. система ведет себя в соответсвии с требованиями во всех предусмотренных ситуациях.

Задача тестирования - определение условий , при которых проявляются дефекты системы.

Так же тестирование помогает экономить ресурсы и дает нашему продукту конкуретные преимущества.

Циклы тестирования ПО.

Основные процессы тестирования ПО:

1. **Планирование**

2. **Мониторинг и контроль**
3. **Проектирование и анализ**
4. **Реализация и выполнение тестов**
5. **Создание отчетности**
6. **Завершение и подведение итогов**

Планирование

1. Определение целей тестирования

Так же проверяем после пройденного цикла что функионал функционирует как и ранее , ошибки исправлены а новые фичи внедрены верно .

2. Решаем сколько ресурсов у нас есть на тестирование .

Мониторинг и контроль

1. Оценка статуса работы
2. Работа с метриками

Обеспечение отслеживаемости процессов и возможность создавать отчеты с конкретными метриками делают процесс тестирования проще и понятнее для всех участников команды разработки.

Проектирование и анализ

1. Рецензирование основы тестирования (целестность , анализ рисков, архитектура, дизайн);
2. Растановка приоритетов ;
3. Подготовка окружения.

На этой стадии мы продумываем какой функционал наше (наше слабое) место, решаем , на что бросить основные силы , подготавливаем тестовые данные, такие как логины и пароли - задача в том , чтобы понять что тестировать и как.

Реализация и выполнение тестов

1. Написание тестовых сценариев;
2. Проведение тестирования.

Пишем тестовые сценарии для нового функционала , проходим тест-кейсы.

Создание отчетности

1. Создание отчетов по тестированию
2. Дополнительные сценарии для проверки

Создаем отчет , в котором описываем, какие ошибки были найдены , какие тесты были успешно пройдены , проверяем отдельные нестандартные сценарии.

Подведение итогов

1. Анализ проведенной работы
2. Выводы о том, как использовать новые знания и как использовать их в будущей работе.

Делаем вывод: например : как нам лучше тестировать новый функционал.

Дополнительные этапы

1. Тестирование сопровождения

2. Работа с отзывами пользователей
3. И др..

Например изучаем отзывы бета-пользователей.

Уровни и типы тестирования

Уровни тестирования определяет то , над чем производится тесты : над отдельным модулем , группой модулей или системой , в целом.

Проведение тестирования на всех уровнях системы - это залог успешной реализации и сдачи проекта.

Фунциональное тестирование

Фунциональное тестирование - рассматривет заранее указанное поведение и основывается на анализе спецификаций функциональности компонента или системы в целом. А проще говоря ,**это тестирование функциональности на соответствие требованиям.**

Виды функционального тестирования :

1. **Компонентный или модульный(Component testing)**
2. **Интеграционный(Integration testing)**
3. **Системный(System testing)**
4. **Приемочный (Acceptance testing)**

Компонентное или модульное

Проверяет функциональность и ищет дефекты в частях приложения , которые доступны и могут быть протестированы по-отдельности(модули программ , объекты,классы,функции и тд.)

Интеграционное

В данном случае **проверяется взаимодействие между компонентами системы после проведения компонентнного тестирования или взаимодействия нашей системы с другими системами.**

Системное

Полностью реализованный программный продукт подвергается системному тестированию. На данном этапе **тестировщика интересует не корректность реализации отдельность процедур и методов , а вся программа в целом , как ее видит конечный пользователь.**

Основой для тестов служат общие требования к программе,включая не только корректность реализации функий , но и производительность , время отклика , устойчивость к сбоям ,атакам, ошибкам пользователя и тд.

Приемочное

Формальный процесс тестирования,который проверяет соответсвие системы требованиям и проводится с целью:

1. определения удолитворяет ли система приемочным критериям
2. вынесения решения заказчиком или другим уполномоченным лицом принимается приложение или нет.

Проводится когда продукт стал соответствовать изначальным требованиям ТЗ.

Нефункциональное тестирование

Нефункционльное тестирование системы выполняется для оценки таких характеристик системы и программного обеспечения , как удобство использования , производительность или безопасность.

Тестирование пользовательского интерфейса

GUI Testing-функциональная проверка интерфейса на соответствие требованиям - размер,шрифт,цвет,верстка.

При тестировании интерфейса :

- **Проверить в разных браузерах**
- **Проверить как он меняется при разном размере экрана**
- **Обращать внимание на ошибки в тексте**
- П**роверить горячие клавишы управления с помощью клавиатуры**
- **Валидации (проверить появляются ли ошибки или подчеркивание поля цветом при неверном вводе)**

Тестирование производительности

Тестированеи производительности (performance testing) производится с целью определить производительность программного продукта.

Тестирование удобства пользования (Usability Testing)

Это метод тестирования , направленный на установление степени удобства использования , обучаемости , понятности и привлекательности для пользователей разрабатываемого продукта в контексте заданных условий.

Тестирование безопасности (safety testing)

Тестирование безопасности : тестирование программного продукта с целью определить его безопасность.

так же :

1. анализ рисков
2. атаки хакеров
3. вирусов
4. несанкционированного доступа к конфиденциальныим данным.

Тестирование установки (installation testing)

Направлено на проверку успешной инсталляции и настройки, а также обновления или удаления программного обеспечения.

Напрмиер , сборка мобильного приложения может прекрасно работать сама ,но если установить ее поверх предыдущей версии из AppStore , она даже не будет запускаться из за внутреннего конфликта.

Тестирование локализаций(Localization Testing)

Вид напрвленный на оценку правильности версии программного продукта с точки зрения языкового и культурного аспекта.

Проверяем работу приложения для определенных регионов,языков , строк , чисел , дат,валют и прочего.

Конфигурационное тестироавние (Configuration Testing)

Специальный вид тестирования , направленный на проверку работы программного обеспечения при различных конфигурациях системы(заявленных платформах,поддерживаемых драйверах, при различных конфигурациях компьютеров и тд)

-Клиентское

-Серверное

Ниже приведены наиболее распространенные типы нефункционального тестирования:

- Тестирование производительности
- Нагрузочное тестирование
- Тестирование отказоустойчивости
- Тестирование совместимости
- Юзабилити-тестирование
- Стресс-тестирование
- Тестирование ремонтопригодности
- Тестирование масштабируемости
- Объемное тестирование
- Тестирование безопасности
- Тестирование аварийного восстановления
- Проверка соответствия
- Тестирование переносимости
- Тестирование эффективности
- Проверка надежности
- Базовое тестирование
- Тест на выносливость
- Тестирование документации
- Тестирование восстановления
- Тестирование интернационализации
- Тестирование локализации

После проведения необходимых изменений, таких как исправление бага/дефекта , программное обеспечение должно быть претестировано для подтверждения того факта , что проблема была действительно решена !

Дымовое тестирование (Smoke Testing)

Дымовое тестирование рассматривается как короткий цикл тестов,выполняемый для подтверждения того , что после сборки кода (нового или исправленного) устанавливаемое приложение запускается и выполняет основные функции.

Регресивное тестирование (Regression Testing)

Это вид тестирования , направленный на проверку изменений, сделанных в приложении или окружающей среде (починка дефекта , слияние кода , миграция на другую операционную систему , базу данных) для подтверждения того факта , что существующая ранее функциональность работает как и прежде.

Тестирование сборки (Build Verification Test)/Дымовое тестирование

Тестирование , направленное на определение соответсвия выпущеной версии, критериям качества для начала тестирвоания.

Санитарное тестирование (Sanity Testing)

Узконаправленное тестирование , достаточное для доказательства того, что функция работает согласно заявленным в спецификации требованиям