***СП***

***Тестирование 5.09***

Silenium

Postman

Библио-Тестирование

Главная и основная цель тестирования, как трудовой деятельности – повышение качества продукта.

Процесс тестирования, в рамках обозначенной цели заключается в поиске, локализации и корректной описании бага.

Согласно теории тестирования – исчерпывающее тестирование невозможно. Невозможно в приложении найти все баги за конечное время. Повышение качества продукта заключается в достижении приемлемой, стабильной работы приложения, устраивающей заказчика и пользователя.

Ресурсы (время, технические ресурсы(номенклатура устройств или программных компонент, которые нам доступны), человеческий ресурс), выделяемые на тестирование, всегда ограничены.

Основная проблема тестирования – необходимость достичь цели, в заданных ограниченных условиях.

Инструменты тестировщика:

Любой таск-менеджер (Редмайн и жира) в этих системах тестировщик фиксирует результат своей работы.

Средства автоматизации – selenium, postman, jemkins и др.

Основным инструментом является инструмент фиксации найденных им багов.

Этапы разработки любого проекта

1. Концепция
2. Договор
3. Техническое задание
4. Дизайн
5. Разработка
6. Стабилизация
7. Релиз
8. Поддержка
9. Закрытие проекта

Тестировщик начинает свою работу на этапе составления тз. Тестировщик тестирует техническое задание на полноту и на непротиворечивость. Под полнотой подразумевается свойство при котором исключается двоякое толкование.

Когда дизайн нарисован, тестировщик проверяет его соответствие требованиям и его полную реализацию.

На этапе разработки тестировщик определяет стратегию и план тестирования, пишет чек-листы, разрабатывает тест-кейсы, может написать автоматизацию на основание требований.

Стратегия тестирования отвечает на вопрос – каким образом в условиях данных ограниченных ресурсов будут подтверждаться заявленные требования. На этапе стратегии определяется вид тестирования и каким образом оно будет проводиться.

***12.09***

Принцип присутствия дефекта – если функционал протестирован, и багов не обнаружено6 это не означает что их нет.

Принцип пестицида – способность багов адаптироваться к существующим тестам. (Вместе с модификацией кода, необходимо модифицировать тесты)

Принцип раннего тестирования – стоимость исправления бага, экспоненциально-пропорционально времени его обнаружения.

Модели разработки:

Итерационная модель разработки.

Весь процесс разработки разделяется на этапы(«спринты»), обычно одинаковой продолжительности, до 4х недель.

В рамках спринта реализуются задачи, взятые из «бэклога», относительно их приоритета.

За время спринта производится полный цикл разработки, относительно задач, которые мы реализуем в рамках этого спринта.

Каждый спринт – маленькая завершенная разработка. После релиза спринта, определяется список(«скоуп») задач, которые либо можно, либо необходимо реализовать.

Водопадная модель разработки.

Весь процесс разработки делится на этапы и к следующему этапу можно переходить только в случае, когда предыдущий этап полностью завершен. Если в процессе реализации какого-то этапа, возникла необходимость изменений на предыдущем этапе, в этом случае производится откат на предыдущий этап, он прорабатывается до полного завершения.

Неоспоримым плюсом этой модели является предсказуемость результата.

Минус – увеличенное время-деньги.

Гибкая модель разработки.

В рамках разработки проекта, все этапы выполняются одновременно. При необходимости внесения изменений, изменения вносятся прямо здесь же.

Неоспоримым плюсом является быстрая скорость разработки, меньшие затраты.

Минус – непредсказуемый результат.

Жизненный цикл тестирования:

1. Общее планирование и анализ требований
2. Уточнение критериев приемки (этап, на котором мы выясняем, что на самом деле хочет заказчик)
3. Уточнение стратегии тестирования
4. Разработка тест-кейсов
5. Выполнение тест-кейсов
6. Фиксация найденных дефектов
7. Анализ результатов тестирования
8. Отчётность

Принцип концентрации багов – 80% багов сконцентрировано в 20% приложений.

Основные места концентрации ошибок:

1. платформа Apple.

2. платежные системы.

3. интеграция с любыми сторонними сервисами.

В требованиях необходимо описывать то, что имеет значение для приложения или проекта в целом.

Источники пути выявления требований:

1. Интервью (проводится с диктофоном, запись прослушивается и выписывается)
2. Работа с фокусными группами
3. Анкетирование (необходимо учитывать, что какой-то процент анкетируемых проставит случайные ответы)
4. Семинары и мозговой штурм
5. Наблюдение (очень важно фиксировать результаты наблюдений)
6. Прототипирование
7. Анализ документов
8. Моделирование процессов и взаимодействие
9. Самостоятельное описание

***19.09 Классификация тестирования***

Упрощённая классификация.

По запуску кода на исполнении:

1. Статическое (без запуска кода
2. Динамическое (с запуском)

По доступу к коду и архитектуре:

1. Метод белого ящика (доступ есть)
2. Чёрного (доступа нету)
3. Серого(частично)

По степени автоматизации:

1. Ручное тестирование
2. Автоматизированное тестирование

По уровню детализации тестирования:

1. Модульное (проверяются отдельные небольшие части приложения)
2. Компонентное
3. Интеграционное (проверяется взаимодействие между несколькими частями приложения)
4. Системное (приложение проверяется как единое целое)

По уровню функционального тестирования:

1. Дымовое тестирование (проверка самой важной, ключевой функциональности, неработоспособность которой делает бессмысленной работу приложения)
2. Тестирование критического пути (проверка функциональности, используемой типичным пользователем, в типичной повседневной деятельности)
3. Расширенное тестирование (проверка всей остальной функциональности, заявленной в требованиях)

По принципам работы с приложением:

1. Позитивное тестирование (Взаимодействие с приложением осуществляется по инструкции, без недопустимых действий, некорректных данных и т. п.
2. Негативное тестирование (В работе с приложением выполняются некорректные операции, некорректные данные, оно не предполагает возникновение багов, предполагает корректную обработку неверных данных)

Подробная классификация тестирования (изъятия):

1. Статическое тестирование (документы, графические прототипы (дизайн), код приложения, параметры, параметры среды исполнения приложения, подготовленные тестовые данные.
2. Ручное тестирование
3. Автоматическое(автоматизированное): Плюсы – существенно быстрей, отсутствие человеческого фактора в процессе тестирование, минимизация затрат, способность генерировать тесты неспособные сделать вручную, способность средства автоматизации собирать, сохранять, автоматизировать и предоставлять в удобном виде всю информацию, способность тестов на низкоуровневой системе. Минусы – необходима более высокая квалификация персонала, высокие затраты на сложные средства автоматизации, на разработку и сопровождение автоматизированных тестов, автоматизированные тест-кейсы необходимо постоянно поддерживать в актуальном состоянии, вплоть до написания новых и удаления старых, автоматизация требует более тщательного планирования и управления рисками, увеличения процентов покрытия кода увеличивает риски, средств автоматизации много

После системного тестирования проводятся альфа-, бета-, гамма-тестирования.

Классификация по природе приложения:

1. Тестирования веб-приложений
2. Тестирование мобильных приложений

Классификация по фокусировке на уровне архитектуры приложения:

1. Тестирование уровня представления (взаимодействие приложения со «внешним миром»)
2. Тестирование уровня бизнес-логики (проверка основного набора функций приложения)
3. Тестирование уровня данных
4. Регрессионное тестирование (тестирование, направленное на проверку того факта, что в ранее работоспособной функциональности не появились ошибки, вызванные изменениями приложения или в среде его функционирования)

ДЗ: расширенная версия классификаций тестирования