

Тестирование с точки зрения тестировщика

Еникеев Р.Р.

О лекции

- Данная лекция является описанием процесса тестирования с точки зрения тестировщика
 - Но некоторые моменты важно знать программисту
- Данная лекция является кратким обзором книги Романа Савина «Тестирование Дот Ком, или Пособие по жестокому обращению с багами в интернет-стартапах»
- Информация из книги не предназначена для тестирования критического ПО

Содержание

- Что такое тестирование, его цели и виды
- Баги и тест-кейсы
- Поддерживаемые тест-кейсы
- Quality Assurance (QA, гарантия качества ПО)
- Место тестирования в цикле разработке ПО
- Стоимость бага
- Цикл тестирования ПО

Баги

- **Баг (bug)** это отклонение фактического результата (ФР, actual result) от ожидаемого результата (ОР, expected result)
 - ФР – результат, который мы получили
 - Добавление товара в корзину не прошло
 - ОР – результат, который мы хотим получить
 - В корзине присутствует выбранный нами товар
- Баг существует когда
 - известен фактический результат,
 - известен ожидаемый результат,
 - известно, что результаты из предыдущих пунктов не совпадают.

Стоимость бага

- Стоимость бага – убытки компании, включающие

- Убытки до передачи ПО:

$$\sum_i^n t_i * c_i, \text{ где}$$

- t_i – время i -го сотрудника на «создание» и исправление бага
- c_i – зарплата i -го сотрудника
- n – количество сотрудников

- Убытки после передачи ПО юзерам

- Отзыв, замена ПО
- Восстановление лояльности клиентов
- Потеря клиентов

- Чем раньше найден баг, тем он будет дешевле для компании

Источники ОР

- Спецификация, спецификация и еще раз спецификация
 - **Спецификация** (spec или **спек**) – детальное описание того, как должно работать ПО, которое составляется *продюсером (PM)*.
- Жизненный опыт, здравый смысл, общение, устоявшиеся стандарты, статистические данные и т.д.

Тестирование

- Самое главное в тестировании – результат.
- **Результат** работы тестировщика – счастье конечного пользователя, связанное с качеством нашего продукта.
- **Тестирование** – это **поиск багов** по алгоритму:
 - узнаем ожидаемый результат,
 - узнаем фактический результат.
 - сравниваем результаты.

Виды багов

- **Функциональный баг** (functional bug) – несоответствие фактической работы кода и спека.
- **Баг в спеке** (spec bug) – несоответствие фактического и ожидаемого содержания спека:
 - Программист, реализующий баг в спеке, не виноват
 - Уведомить продюсера (при этом еще и письменно по почте, сохраняя всю переписку, чтобы в случае чего прикрыть себя – бизнес есть бизнес).

Примеры багов в спеке

- Нет четкого описания ошибки при вводе неправильного пароля
 - «Должно выводиться сообщение ошибки» вместо
 - «Должно выводиться «Вы ввели неверный пароль»»
- При попытке доказать, что «я не робот» не говорится, нужно ли выбрать краешек дорожного знака или нет

Цель тестирования

- **Цель тестирования** – нахождение багов до того, как их найдут пользователи.
- 100% тестирование ПО **невозможно** из-за нехватки ресурсов, поэтому просочившиеся баги – это неизбежное зло, количество которых тестировщик должен свести к минимуму.
- Баги бывают разных приоритетов, которые отражают важность бага (по аналогии с фичами для реализации программистами).
- Данные о найденных багах можно анализировать статистически и посмотреть кто виноват. Анализ статистики это уже QA.

Тестирование и QA (Quality Assurance)

- Тестирование и QA это забота о качестве ПО, но
 - QA (гарантия качества) в виде превентирования появления багов через улучшение процесса разработки ПО,
 - Тестирование в виде обнаружения (поиска) багов до того, как их найдут пользователи.
- Тестировщик получает ПО уже с багами и поэтому не является единственным, кто отвечает за качество продукта
- Качество – это результат
 - работы всех участников процесса разработки ПО,
 - отлаженности и настроек всего процесса.

Тест-кейс (test case)

- **Тест-кейсы** – документация по исполнению тестирования:
 - ожидаемый результат,
 - шаги, которые должны привести к ФР (инструкция исполнения).
- Тест-кейс – аналог спека для программиста
- Хранится в CVS

Идея: Покупка в интернет магазине

Шаги:

1. Открыть страницу
2. Войти в аккаунт “а” с паролем “1”
3. Выбрать товар “b”
4. Поместить в корзину
5. Оплатить товар кредитной картой “123”

ОР:

1. Создан заказ, содержащий товар “b”
2. Должна быть произведена оплата с карты “123”

Создание и исполнение тест-кейса

- Создание тест-кейса (test case generation) – процесс придумыва-ния и написания тест-кейсов
- Исполнение тест-кейса (test case execution)
 - Выполнение шагов тест-кейса
 - Сравнение ОР и ФР
- Исход исполнения тест-кейса (test case result)
 - Исполнение тест-кейса завершено
 - Положительный исход (**PASS**), если ОР равен ФР.
 - Отрицательный результат (**FAIL**), если ОР не равен ФР.
 - Мы заблокированы (test execution is blocked) – мы не можем пройти все шаги тест-кейса.

Атрибуты тест-кейса

- Уникальный id – дается один раз и не меняется.
- Приоритет – полезен при отборе тест-кейсов для тестирования.
- Идея – краткое описание того, что мы тестируем. (Тест-кейс могут прочитать через месяц).
- Подготовительная часть
 - данные об аккаунте пользователя,
 - данные о кредитке и
 - другие вещи, облегчающие исполнение и поддержку тест-кейса.
- История редактирования, где отражается «Кто? Что? Зачем? Когда? Почему?»

Поддерживаемые (maintainable) тест-кейсы

Поддерживаемость

1. Сделать тест-кейс *data-driven* – разделить и слинковать данные и инструкции по их применению
2. Не описывать шаги по явно очевидным сценариям
3. Не давать конкретных деталей, если они не играют роли при исполнении тест-кейса
4. Вынести во внешний документ повторяющиеся сценарии (шаги оплаты)

Идея: Покупка в интернет магазине

Подготовительная часть:

username: a
password: 1
credit card: 123

Шаги:

1. Открыть страницу
2. Войти в аккаунт, используя данные из ПЧ
3. Выбрать произвольный товар
4. Поместить в корзину
5. Оплатить товар кредитной картой из ПЧ

ОР:

1. Создан заказ, содержащий выбранный товар
2. Должна быть произведена оплата с исп-ой карты

Качественные тест-кейсы

- Независимы – не связан с другими тест-кейсами, т.к.
 - Тест-кейсы могут быть изменены/удалены
 - Запущены в произвольном порядке
 - Независимы тест-кейс должен удовлетворять условиям
 - отсутствие ссылок на другие тест-кейсы;
 - независимость от «следов», оставленных другими тест-кейсами.
 - Копирование частей тест-кейса может быть не очень хорошая, но преимущества независимого тест-кейса может перекрыть это неудобство.
- Четкая и ясная формулировка шагов
 - То, что очевидно сейчас, может быть неочевидным в будущем
 - Ваши тест-кейсы будут читать другие люди
 - Излишняя детализация ведет к усложнению поддерживаемости
- Четкая формулировка идеи и/или ОР. Пример плохого ОР
 - «На экране должна быть выведена ошибка»
 - «Все работает»

Количество ожидаемых результатов

- Теоретически: тест-кейсом должен проверяться только один ОР
- На практике
 - Бывает проверка двух ОР, тогда существует 2 решения
 - Разложить тест-кейс на два
 - Оставить один тест-кейс. Тест-кейс будет иметь положительный результат, только если оба фактических результата совпадут с соответствующими ОР.
 - В тест-кейсе проверяет одну идею (логический ОР)
- Два ОР может возникнуть при тестировании фронт-енда и бэк-энда.

Тест-комплект (test case suite)

- Тест-кейс проверяет одну вещь.
- **Тест-комплект** – совокупность тест-кейсов по определенному критерию, которые, например, проверяют
 - определенную часть проекта («оплата») и/или
 - определенный спек.
- Можно запускать весь тест-комплект сразу

Классификация и виды тестирования

- Классификация подразумевает критерий разбиения
- Признаки классификации
 - По знанию внутренней реализации
 - По объекту тестирования
 - По времени проведения тестирования
 - По критерию «позитивности» сценариев
 - По степени изолированности тестируемых компонентов
 - По степени автоматизированности тестирования
 - По степени подготовки к тестированию

По знанию внутренней реализации

- Черный ящик (black box testing) - незнание реализации
 - Тестирование – ввод данных и получение фактического результата
 - Идеи для тестирования берутся из паттернов поведения пользователя
- Белый ящик (white box testing) – знание реализации
 - Тестируем определенную часть внутренней реализации (бэкэнда)
- Серый ящик (grey box testing) – объединение двух предыдущих
 - Пример: тестирование регистрации пользователя
 - Проверяем сообщение на сайте (черный ящик)
 - Проверяем создание записи в БД (белый ящик)

По объекту тестирования

- Функциональное тестирование (functional testing) – проверка работы функциональностей
 - **Функциональность (functionality, feature)** – средство для решения некой задачи
- Тестирование скорости и надежности (performance/load/stress testing)
 - Есть специальные инструменты
- Тестирование совместимости (compatibility testing) – как наше ПО взаимодействует с компьютером (железо, ОС, ...) пользователя
- Тестирование локализации (localization testing) – проверка работы ПО в разных странах (разные языки)
- Тестирование интерфейса пользователя (UI testing)
 - Пример: не съехала ли верстка, корректность надписей на кнопкой, ...
- Тестирование удобства использования (usability testing)
- Тестирование безопасности (security testing)

По времени проведения тестирования

- До передачи пользователю (альфа-тестирование)
 - Тест приемки (smoke test) – самое простое тестирование (поверхностное)
 - Тестирование новых функциональностей
 - Регрессивное (regression testing) – тестируем программу
 - Тест сдачи (acceptance testing) – перед релизом
- После передачи пользователю (бета-тестирование)

По критерию «ПОЗИТИВНОСТИ» сценариев

- Позитивное тестирование (positive testing) – предполагает «правильное» использование и/или работу системы.
- Негативное тестирование (negative testing)
 - Проверяется
 - Потенциальная ошибка пользователя (error)
 - Ввод строки вместо числа
 - Ввод некорректного email'а
 - Потенциальный дефект (failure) в системе
 - Отсутствуют/некорректные файлы, используемые ПО
 - Отсутствие подключения к интернету
 - Находит больше ошибок, чем позитивное
- Позитивные тесты исполняются в первую очередь

По степени изолированности тестируемых компонентов

- Компонентное (component)
- Интеграционное (integration)
- Системное (system or end-to-end) – лучше производить вначале

	Уровень тестирования	Объект тестирования
Компонентное	1 логический компонент	Сам логический компонент
Интеграционное	Два и более логических компонента	Взаимодействие компонент
Системное		Вся система от начала до конца

Остальные виды

- По степени автоматизированности тестирования
 - Ручное
 - Автоматизированное
 - Полуавтоматизированное
- По степени подготовки к тестированию
 - Тестирование по документации (по тест-кейсам)
 - Интуитивное/эд-хок тестирование (ad-hoc testing)

Цикл разработки ПО и тестирование

- Цикл (процесс) разработки ПО – это путь от идеи до поддержки готового ПО:
 - *Идея*
 - *Разработка дизайна продукта и создание документации*
 - *Кодирование*
 - *Исполнение тестирования и ремонт багов*
 - *Релиз*

Этап «Идея»

- Идея – это описание ЦЕЛИ, а дизайн – это описание ПУТИ к достижению этой цели.
- Результат этапа – документ о требованиях маркетинга

Этап «Создание документации»

- Результат данного этапа – спек, содержащий уникальные имя и название, приоритет
- Спек должен удовлетворять условиям
 - Акцент на деталях и их четкое определение.
 - Забота о недопущении неверного толкования.
 - Непротиворечивость внутри спека и с другими спеками.
- Спекы могут меняться, поэтому необходимо предотвратить изменение спека
 - К определенной дате спекы замораживаются
 - Изменение спека возможно только с помощью процедуры изменения спека, в которой участвуют все участники процесса разработки.

Неверное толкование спека

- Неверное толкование спека программистом или тестером, написанного продюссером – частая ошибка
- Тестировщики должны настаивать на создании БМП в дополнение спека:
 - Блок-схемы – алгоритм высокой степени абстракции,
 - Макеты (эскизы) – представляют интерфейс пользователя с достаточной степенью детализации,
 - Примеры.
- Использование БМП ведет
 - К адекватному толкованию спека разными людьми благодаря описанию предмета с разных сторон (превентирование багов)
 - Нахождению ошибок в спеках

Этап «Кодирование»

- Тестировщики планируют проверку пишущегося кода.
- Результат тестировщиков на данном этапе – тест-кейсы
- Перед началом тестирования убедитесь:
 - Код заморожен (аналог заморозки спеков перед этапом «кодирование»)
 - Версия продукта является той, которую вы должны протестировать.
- Полезно выполнить рассмотрение тест-кейсов (test case review):
 - Изучаются тест-кейсы (аналог code review)
 - Участники – продюсер, программист и тестировщик
 - Продюсеры или программисты дают новые идеи для тестирования

Превентирование багов на этапе «Кодирование»

- Наличие требований к содержанию спеков
- Возможность быстрой коммуникации между всеми участниками
 - Если к тебе обратились – помоги
 - Устное общение лучше подтверждать e-мейлом (бизнес есть бизнес)
- Code review и стандарты программирования
- Реальные сроки
 - Баланс между тем, чтобы закончить в срок и качеством
 - Лучше сделать меньше, но качественнее
- Требования к проведению юнит-тестирования
- Финансовые рычаги для написания эффективного и «чистого» кода
- «Качество» и «счастье пользователя» – основа корпоративной философии

Этап «Исполнение тестирования и ремонт багов»

- Тест приемки (smoke test) – проверка основных функциональностей
- тестирование новых функциональностей (new feature testing)
- регрессивное тестирование (regression testing).

Большая картина цикла разработки

Участник	Роль	Стадия
Маркетолог	Генерирует идеи и составляет MRD	Идея
Продюсер	Разрабатывает и документирует дизайн продукта	Дизайн и документация
Программист	Переводит дизайн продукта на ЯП	Кодирование
	Ремонтирует баги	Тест и ремонт
Тестировщик	Готовится к исполнению тестирования	Кодирование
	Исполняет тестирование	Тест и ремонт

Цикл тестирования ПО

- Цикл тестирования состоит из трех этапов
 1. Изучение и анализ предмета тестирования. Цели этапа - понять
 - какие функциональности предстоит протестировать
 - как эти функциональности работают
 2. Планирование тестирования.

Задача – найти компромисс между объемом тестирования, который возможен в теории и на практике.
 3. Исполнение тестирования – поиск багов в ПО с использованием созданных тест-кейсов.
- Первые два этапа часто объединяют в «Подготовка к тестированию» (test preparation)

При нахождении бага

- Баг может быть найден в любой момент
- После нахождения бага тестировщик заносит запись о нем в систему трэкинга багов (СТБ)
 - Программист также может сообщить о баге
- После починки бага (в коде программистом или в спеке продюсером), тестировщик проверяет:
 - **действительно ли баг был починен** – регрессивное тестирование (bug regression testing),
 - Не появились ли новые баги.

Подготовка к тестированию

- Самое главное – ментальный настрой:
 - У тестировщика – деструктивное мышление (на разрушение)
Код – это убежище багов
 - У программиста – конструктивный (на созидание)
- Порядок тестирования
 - Позитивное
 - Негативное
- Подготовка к тестированию
 - Создание или изменение тест-кейсов
 - Отбор тестов
 - Оценка риска (risk estimation)
 - эквивалентные классы (equivalent classes)
 - пограничные значения (boundary values)

Подготовка к регрессивному тестированию

- Тест-комплект для тестирования выбирается исходя из факторов:
 - К какой части ПО принадлежат новые фичи
 - Какие старые фичи напрямую зависят от части ПО с новыми фичами.
- Решение проблемы противоречия между ограниченными ресурсами и увеличивающимся количеством тест-комплектов.
 - Приоритизация тест-комплектов и тест-кейсов.
 - Оптимизация тест-комплектов (Уменьшение количества тест-кейсов)
 - Автоматизация регрессивного тестирования

Ключевые идеи раздела

- Цель тестирования – счастье клиентов ПО
- Тестирование необходимо для экономии денег компании и поиск багов необходимо производить как можно раньше
- Важность правильной спецификации
- Поддерживаемость тестов (думаем о будущем)
- Деструктивный ментальный настрой тестировщика
- Важность негативного тестирования
- За качество отвечают все участники процесса разработки
- Бизнес есть бизнес – страхуйте свою спину от чужих ошибок