Технологии программирования. Тестирование.

весна 2023

Анекдот

Заходит однажды тестировщик в бар.

Забегает в бар.

Пролезает в бар.

Танцуя, проникает в бар.

Крадется в бар.

Врывается в бар.

Прыгает в бар.

Заказывает:

кружку пива,

2 кружки пива,

0 кружек пива,

999999999 кружек пива,

ящерицу в стакане,

–1 кружку пива,

qwerty кружек пива.

Первый реальный клиент заходит в бар и спрашивает, где туалет. Бар вспыхивает пламенем, все погибают.

Agenda

- Что такое тестирование
- Основные классификации
- Что делают тестировщики
- Что делают разработчики

Что такое тестирование?

“Классическое” определение:

Тестирование - проверка соответствия между реальным и ожидаемым поведением на конечном наборе тестов, выбранных определённым образом.

Альтернативное определение:

Тестирование - вероятностный процесс нахождения ошибок.

Несмотря на то, что проводится тестирование, ошибки могут быть не выявлены и оказаться в производственной эксплуатации.

Фактически в тестах проверяется, что система работает для конкретного набора условий. У пользователей условия могут оказаться иными.

Особенно, если пользователи - злоумышленники.

Процесс

В развивающемся продукте тестированием нужно заниматься постоянно, это не разовая деятельность, повторяется непрерывно.

Принятие решения по выявленным ошибкам

Градация серъёзности ошибок, принятая в компании.

Записать в систему отслеживания ошибок (bug tracker)?

Остановить выпуск очередной версии продукта (release)?

Устранить на месте?

…

Зачем нужно тестирование?

Позволяет сделать продукт качественным

Упрощает и удешевляет процесс разработки, так как сокращает время между “созданием” ошибки и её обнаружением

ГОСТ Р 56920-2016/ISO/IEC/IEEE 29119

Серия стандартов ИСО/МЭК/ИИЭР 29119 под общим названием "Системная и программная инженерия. Тестирование программного обеспечения".

Цель создания серии стандартов ИСО/МЭК/ИИЭР 29119 "Тестирование программного обеспечения" состоит в том, чтобы определить на международном уровне согласованную совокупность стандартов, которая может использоваться любой организацией при выполнении различных форм тестирования программного обеспечения.

Классификации и виды тестирования

- Функциональное / нефункциональное
- Ручное / автоматическое
- Основное / smoke / регрессионное
- Белый ящик / чёрный / серый
- Unit / интеграционное / E2E
-

Функциональное / нефункциональное

Функциональное тестирование выполняется чтобы убедиться, что каждая функция программного приложения ведет себя так, как указано в документе с требованиями.

Нефункциональное тестирование проводится для проверки нефункциональных требований приложения, таких как производительность, безопасность, совместимость, надежность, удобство использования и т. д

Нагрузочное тестирование (load testing)

Тип тестирования уровня производительности, проводимого для оценки поведения элемента тестирования при ожидаемых условиях переменной нагрузки, обычно для ожидаемых условий низкого, типичного и пикового использования.

Выполняется для диагностики поведения системы при увеличении рабочей нагрузки.

Стрессовое тестирование (Stress testing)

Тип тестирования уровня производительности, проводимого для оценки поведения элемента тестирования при условиях загрузки, выше ожидаемой или указанной в требованиях к производительности, или при доступности ресурсов, ниже минимальной, указанной в требованиях.

Ручное / автоматическое

Ручное тестирование лучше подходит для проверки небольших изменений. Во время ручного тестирования тестировщики часто могут найти такие проблемы, которые остались бы незамеченными, если бы они полагались только на автоматизированные тесты. Ручное тестирование не требует глубоких знаний языков программирования.

При работе над большими приложениями, тестирование без использования автоматических тестов будет слишком долгим и дорогим.

Дымовые тесты (smoke tests)

Предназначены для проверки базовой функциональности приложения. Это быстро выполнимые тесты, с помощью которых тестировщики следят за тем, чтобы основные функции системы работали правильно.

Тестированием занимаются

- **разработчики**,
- специалисты по безопасности,
- специально обученные люди - **QA** (quality assurance),
- …

Основные этапы QA процесса

- Работа с требованиями
- Планирование тестирования
- Разработка тестовых сценариев
- Тестирование программного обеспечения
- Повторное тестирование
- “Завершение” тестирования

Работа с требованиями

Процесс QA начинается на самых ранних этапах жизненного цикла разработки программного обеспечения — на этапе анализа требований.

Тестировщики проверяют требования и функциональные спецификации, чтобы убедиться, что они чёткие, непротиворечивые, полные, выполнимые и их возможно протестировать.

Сотрудничают с бизнес-аналитиками.

Планирование тестирования

После изучения требований,тестировщики разрабатывают стратегию тестирования и планирование процедур по контролю качества.

На этом этапе определяют объем работ и бюджет, решают, какой подход использовать на каждом этапе разработки программного обеспечения, какие виды и типы тестирования потребуются, какие инструменты лучше использовать.

Разработка тестовых сценариев

Тестировщики

- разрабатывают тестовые сценарии (тест кейсы),
- создают чек-листы,
- подготавливают среду для выполнения тестов,
- создают сценарии для автоматического тестирования.

Сотрудничают с разработчиками.

Основное тестирование

Производится поиск ошибок и дефектов.

Выполняются различные типы тестов.

Тестировщики сообщают обо всех обнаруженных ошибках, принимаются решения.

Регрессионное тестирование (повторное)

Тестирование уже протестированной программы, проводящееся после модификации для уверенности в том, что процесс модификации не внес или не активизировал ошибки в областях, не подвергавшихся изменениям. Проводится после изменений в коде программного продукта или его окружении.

Завершение тестирования

Подготовка отчёта о результатах тестирования.

В документации описываются все тесты, выполненные в течение жизненного цикла разработки программного обеспечения.

Работа над ошибками и так далее…

Тестирование, проводимое разработчиками

Автоматические тесты, встроенные в процесс непрерывной интеграции (CI = continuous integration)

Test Driven Development (TDD)

Test Driven Development (TDD)

Разработка на основе тестов основывается на повторении коротких циклов разработки: изначально пишется тест, покрывающий желаемое изменение, затем пишется программный код, который реализует желаемое поведение системы и позволит пройти написанный тест. Затем проводится рефакторинг написанного кода с постоянной проверкой прохождения тестов.

Модульное (unit) тестирование

- Тестирует небольшой фрагмент кода
- Делает это быстро
- Изолирован от других тестов

Как правило, unit-тесты не зависят от фреймворков, баз данных и внешних сервисов. Используются заглушки = mock-объекты или stub-объекты - псевдореализация необходимых зависимостей.

Интеграционное тестирование

Используется

- для проверки совместной работы нескольких частей системы
- для проверки работы с внешними системами

Сквозное тестирование (end-to-end)

- Полностью развёрнутая система
- Проверяется взаимодействие между сервисами
- Тесты работают поверх API или интерфейса
- Дорого

“Соседи”

- Статический анализ кода
- Тестирование производительности
- Тестирование безопасности
- Тестирование надёжности
- Мониторинг

Что проверять в тестах

- Основные сценарии
- Граничные случаи
- Сценарии с ошибками (ожидаемыми и неожиданными)
- Специфику функциональности (параллельное выполнение, транзакционность, …)

Эффективные тесты

- Ловят баги
- Устойчивы к рефакторингу
- Просто читаются
- Просто запускаются
- Быстро работают
- Встроены в процесс разработки

Процент покрытия тестами

Code coverage - метрика, показывающая процент кода, покрытого тестами.

Хорошая практика - покрытие больше 70%.

100% покрытие не гарантирует отсутствие ошибок.

Можно использовать показатель при настройке CI.

Code style для тестов

- Качество кода тестов должно быть не хуже основного.
- Тесты должны хорошо читаться. Единые правила именования и единообразная структура.
- Комментарии в тестах.
- Набор простых тестов лучше одного сложного. Причина падения должна быть понятна.
- Параллельное выполнение тестов важно для больших проектов.

Что сложно тестировать

Параллельные потоки и окрестности

Сервисы, эксплуатирующие случайность

“В сухом остатке”

Необходим разумный баланс в трудозатратах на тестирование.

Грамотное тестирование при разработке больших систем улучшает качество кода, делает разработку проще, быстрее, дешевле.

Что почитать

Книга:

«Тестирование программного обеспечения. Базовый курс.»

Автор: Куликов Святослав Святославович

https://svyatoslav.biz/software_testing_book/

Копилка материалов по QA Владислава Еремеева

https://github.com/VladislavEremeev/QA_bible

Атрибуты требований

Корректность — точное описание разрабатываемого функционала.

Проверяемость — формулировка требований таким образом, чтобы можно было выставить однозначный вердикт, выполнено все в соответствии с требованиями или нет.

Полнота — в требовании должна содержаться вся необходимая для реализации функциональности информация.

Недвусмысленность — требование должно содержать однозначные формулировки.

Модифицируемость — в каждое требование можно внести изменение.

Атрибуты требований (продолжение)

Непротиворечивость — требование не должно содержать внутренних противоречий и противоречий другим требованиям и документам.

Приоритетность — у каждого требования должен быть приоритет (количественная оценка степени значимости требования). Этот атрибут позволит грамотно управлять ресурсами на проекте.

Атомарность — требование нельзя разбить на отдельные части без потери деталей.

Прослеживаемость — каждое требование должно иметь уникальный идентификатор, по которому на него можно сослаться.