Manual QA course

Lecture 8. Виды тестирования. Часть 3

Дорофеев Максим

hillel
компьютерная школа

Виды тестирования. По степени подготовленности к тестированию.

- Тестирование по документации (**formal testing**);
- Интуитивное тестирование (**ad hoc testing**);
- Исследовательское тестирование (**Exploratory testing**).

Интуитивное тестирование.

Выполняется:

- При полном отсутствии плана и документации;
- С использованием собственного опыта и здравого смысла.

Интуитивное тестирование. Ad - hoc testing.

Плюсы:

- Находятся “хитрые и коварные” дефекты;
- Нет затрат на проектирование тестов;
- Ускоряется обучение новых сотрудников;
- Легкость в осуществлении.

Интуитивное тестирование. Ad - hoc testing.

Минусы:

- Нет гарантий по покрытию тестами;
- Высокий риск пропустить ошибку в стандартных функциях

Исследовательское тестирование. Exploratory testing

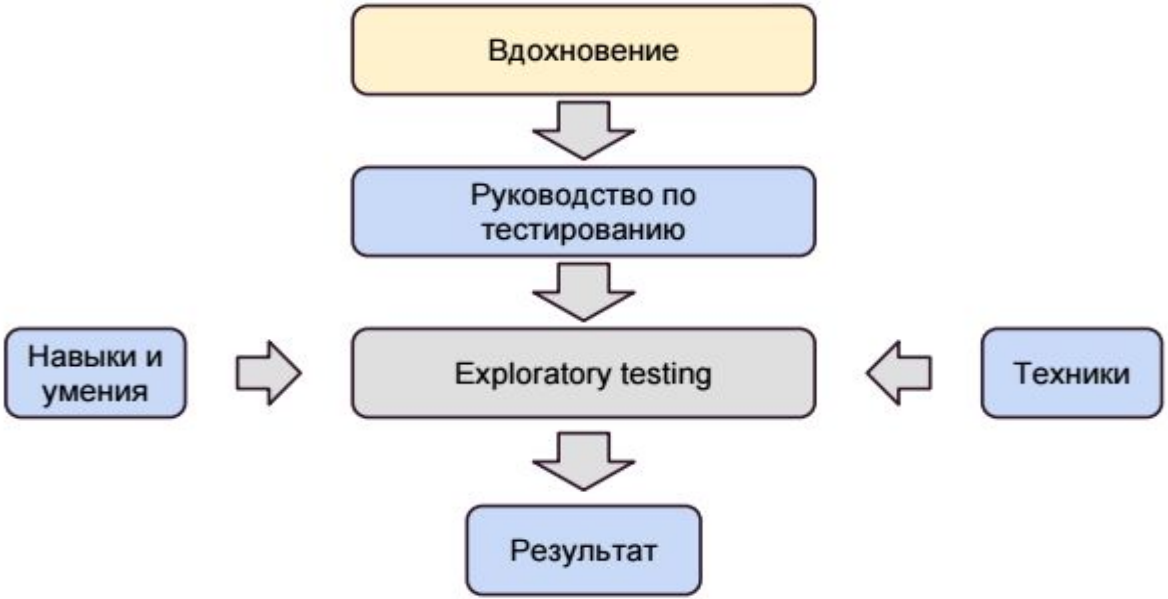
Скорее подход, чем вид тестирования.

Исследовательское тестирование. Exploratory testing

- более формальная версия Ad - hoc тестирования, не требующая написания тест - кейсов, но подразумевающая, что каждый последующий тест выбирается на основе результатов предыдущего.

Исследовательское тестирование. Exploratory testing

Исследовательское тестирование. Exploratory testing



Исследовательское тестирование. Вдохновение.

1. Информация
   - Книги;
   - Сайты;
   - Документация по продукту.
2. Модель
   - Сценарии использования;
   - Требования;
   - Функционал.

Исследовательское тестирование. Руководство.

- Идеи;
- Чеклисты;
- Особенности функционирования;
- Перечень рисков;
- Покрытие.

Исследовательское тестирование. Результаты.

- Баг - репорты;
- Заметки;
- Отчёты о состоянии ПО;
- Другое.

Исследовательское тестирование. Плюсы.

- Возможность найти больше дефектов;
- Не нужно тратить время на предварительное описание всех сценариев;
- Не нужна поддержка тестовых сценариев;
- Не происходит привыкание к тестовым сценариям;
- Не теряется цельное видение продукта;
- Тестирование проходит быстрее;

Исследовательское тестирование. Когда применять?

- Мало времени;
- Сложности с требованиями;
- Небольшой проект;
- Пришел внезапный запрос на изменения;
- Тестировщики постоянно проходят одни и те же сценарии;
- Когда хочется перестраховаться.

Исследовательское тестирование. Когда одним исследовательским тестированием не обойтись.

- Приложение стандартизировано;
- Проводится интеграционное тестирование;
- Тестовые сценарии отдаются на аутсорс;
- Длительный проект.

Исследовательское тестирование. Мифы.

- Исследовательское тестирование невозможно проконтролировать.

Исследовательское тестирование. Мифы.

- Нельзя доверить тестирование первому встречному.

Исследовательское тестирование. Мифы.

- Сложно “продать” исследовательское тестирование заказчику, объяснить его необходимость.

Исследовательское тестирование. Что не есть Exploratory Testing?

Заблуждение «Быстрые проверки – это и есть исследовательское тестирование».

Исследовательское тестирование. Что не есть Exploratory Testing?

Заблуждение «Exploratory testing – это недокументированный процесс».

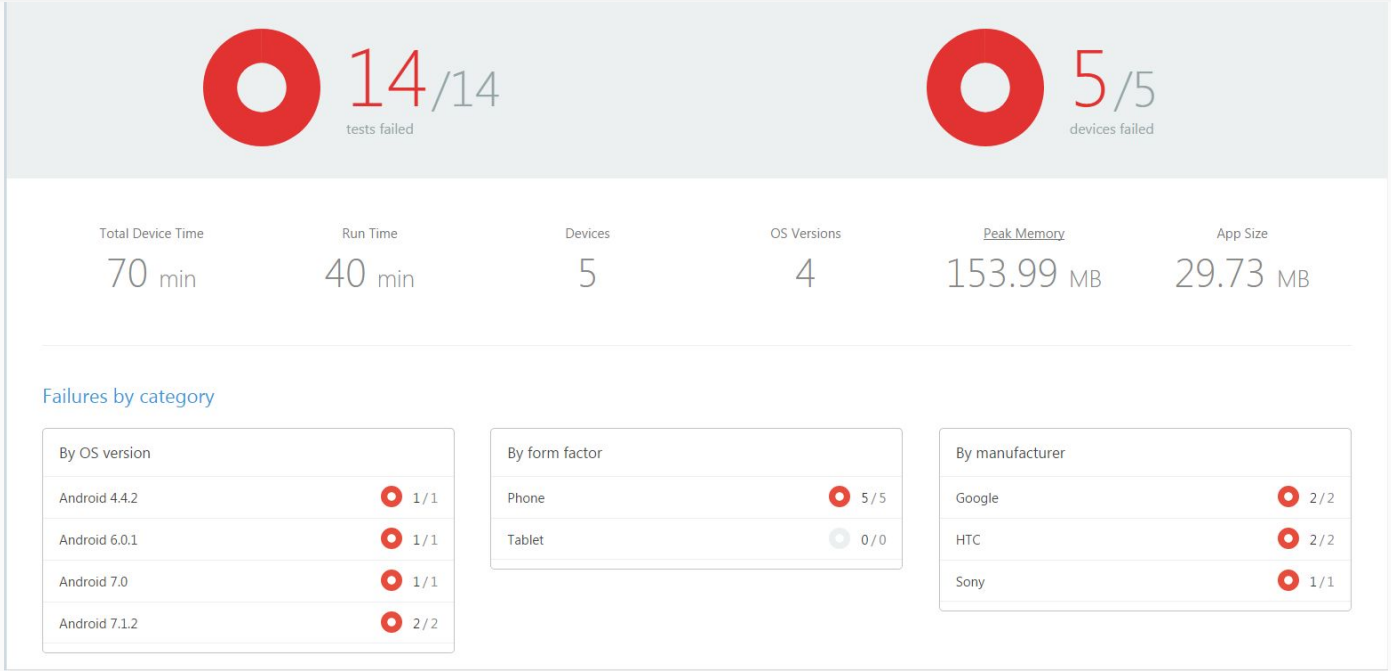
Исследовательское тестирование. Выводы.

- Исследовательское тестирование - не означает полное отсутствие документации и хаос;
- Комбинируя типы тестирования можно подобрать необходимый уровень документации для проекта;
- Сценарное и исследовательское тестирование компенсируют недостатки друг друга.

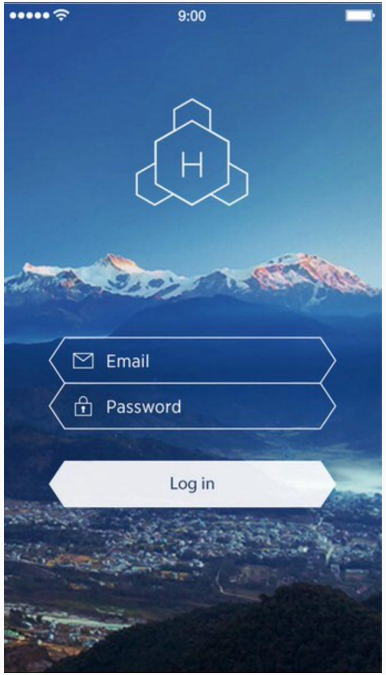
Тестирование GUI. Заблуждения.

- Пользовательский интерфейс - самая простая часть проекта;
- Главное, чтобы работало, а как выглядит - неважно;
- Это никто не заметит;
- Оттестируем и исправим, когда будет время.

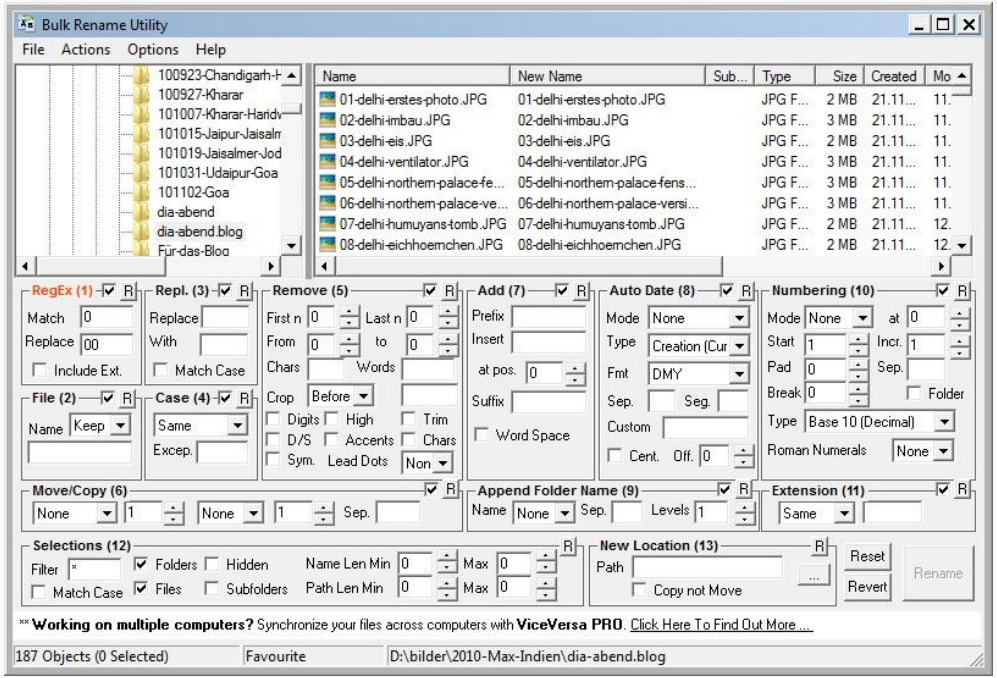
Тестирование GUI. Хороший пример. WEB.



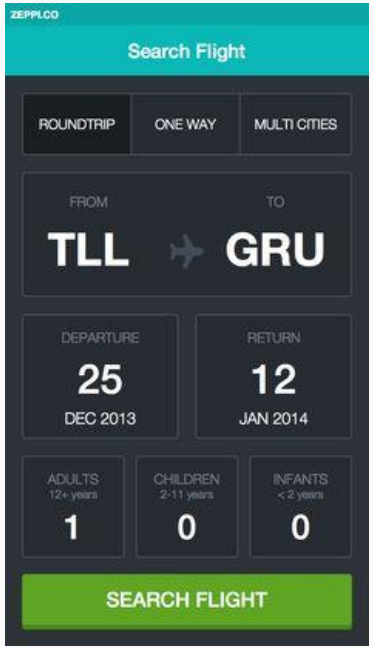
Тестирование GUI. Хороший пример. Mobile.



Тестирование GUI. Плохой пример. WEB.



Тестирование GUI. Плохой пример. Mobile



Тестирование GUI. Задачи.

- Ошибки в функциональности посредством интерфейса;
- Необработанные исключения при взаимодействии с интерфейсом;
- Потеря или искажение данных, передаваемых через элементы интерфейса;
- Ошибки в интерфейсе (несоответствие проектной документации, отсутствие элементов интерфейса).

Тестирование GUI. Фазы.

- Анализ требований к пользовательскому интерфейсу;
- Разработка документации;
- Выполнение тестов и сбор информации;
- определение полноты покрытия пользовательского интерфейса требованиями;
- Предоставление информации о этапе тестирования.

Тестирование GUI. Методы

- Ручное тестирование;
- Автоматическое тестирование.

Тестирование GUI. Ручное тестирование.

Плюсы:

- Поиск “Косметических” дефектов;
- Анализ выполняется по формальным признакам, а согласно человеческому восприятию.

Тестирование GUI. Ручное тестирование.

Минусы:

- Требуются значительные человеческие и временные ресурсы;
- При проведении повторных циклов тестирования, время затраченное на тестирование возрастает.

Тестирование GUI. автоматизированное тестирование.

Плюсы:

- Высокая скорость выполнения;
- больший объем покрытия;
- Не требуется участие людей.

Тестирование GUI. Автоматизированное тестирование.

Минусы:

- Анализ успешности будет выполнятся по формальным признакам;
- Невозможность поиска косметических дефектов;
- Высокая стоимость поддержки.

Interoperability Testing.

Тестирование взаимодействия - это функциональное тестирование, проверяющее способность приложения взаимодействовать с одним и более компонентами или системами и включающее в себя интеграционное тестирование (**integration testing**)

Виды тестирования связанные с изменениями.

- Дымовое тестирование (Smoke Testing);
- Регрессионное тестирование (Regression Testing);
- Тестирование сборки (Build Verification Test);
- Санитарное тестирование или проверка согласованности/исправности (Sanity Testing).

Виды тестирования связанные с изменениями. Smoke testing

Понятие дымовое тестирование пошло из инженерной среды:

"При вводе в эксплуатацию нового оборудования ("железа") считалось, что тестирование прошло удачно, если из установки не пошел дым"

Smoke testing.

В оригинальном своем применении smoke - тестирование предназначено для проверки самых простых и очевидных кейсов, без которой любой другой вид тестирования будет неоправданно излишним.

Smoke testing. Примеры тестов.

- Функция входа в систему;
- Функции связанные с управлением данных (Запись, хранение, обработка, удаление, изменение и тд.);
- Функции связанные с доступом ко всем вкладкам.

Smoke testing.

Для составления набора smoke - тестов, необходимо определить какие задачи нужно решить нашему приложению, какие важные требования мы должны соблюдать и в какой последовательности.

Smoke testing.

Вывод о работоспособности основных функций делается на основании результатов поверхностного тестирования наиболее важных модулей приложения на предмет возможности выполнения требуемых задач и наличия быстронаходимых критических и блокирующих дефектов. В случае отсутствия таковых дефектов дымовое тестирование объявляется пройденным, и приложение передается для проведения полного цикла тестирования, в противном случае, дымовое тестирование объявляется проваленным, и приложение уходит на доработку.

Smoke testing.

Аналогами дымового тестирования являются Build Verification Testing и Acceptance Testing, выполняемые на функциональном уровне командой тестирования, по результатам которых делается вывод о том, принимается или нет установленная версия программного обеспечения в тестирование, эксплуатацию или на поставку заказчику.

Smoke testing.

Smoke - тесты - самые первые кандидаты на автоматизацию!

Sanity Testing.

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

Sanity testing. Особенности.

- Глубокое исследование определенной функциональности приложения.
- Это как правило ручное тестирование (Не лучший кандидат для автоматизации);
- Проверка работы функции (модуля) в соответствии требованиям (спецификациям);
- Это своего рода приемочное тестирование.

Sanity Testing vs Smoke Testing

Эти виды тестирования имеют "вектора движения", направления в разные стороны. В отличии от дымового (**Smoke testing**), санитарное тестирование (**Sanity testing**) направлено **вглубь проверяемой функции**, в то время как дымовое направлено **вширь**, для покрытия тестами как можно большего функционала в кратчайшие сроки.

Regression Testing.

Вид тестирования направленный на проверку изменений, сделанных в приложении или окружающей среде (починка дефекта, слияние кода, миграция на другую операционную систему, базу данных, веб сервер или сервер приложения), для подтверждения того факта, что существующая ранее функциональность работает как и прежде. Регрессионными могут быть как функциональные, так и нефункциональные тесты

Regression Testing.

Как правило, для регрессионного тестирования используются тест - кейсы, написанные на ранних стадиях разработки и тестирования. Это дает гарантию того, что изменения в новой версии приложения не повредили уже существующую функциональность. Рекомендуется делать автоматизацию регрессионных тестов, для ускорения последующего процесса тестирования и обнаружения дефектов на ранних стадиях разработки программного обеспечения.

Regression Testing.

3 основных типа регрессионного тестирования:

- Регрессия багов (**Bug regression**) - попытка доказать, что исправленная ошибка на самом деле не исправлена
- Регрессия старых багов (**Old bugs regression**) - попытка доказать, что недавнее изменение кода или данных сломало исправление старых ошибок, т.е. старые баги стали снова воспроизводиться.
- Регрессия побочного эффекта (**Side effect regression**) - попытка доказать, что недавнее изменение кода или данных сломало другие части разрабатываемого приложения

Build Verification Test.

Тестирование направленное на определение соответствия, выпущенной версии, критериям качества для начала тестирования. По своим целям является аналогом Smoke testing, направленного на приемку новой версии в дальнейшее тестирование или эксплуатацию. Вглубь оно может проникать дальше, в зависимости от требований к качеству выпущенной версии

Build Verification Test.

При установке новой версии сборки, команда тестирования должна приступить к приемочному тестированию этой сборки.

Команда должна за максимально короткое время проверить наибольшее количество функционала (ручными и автоматизированными тестами).

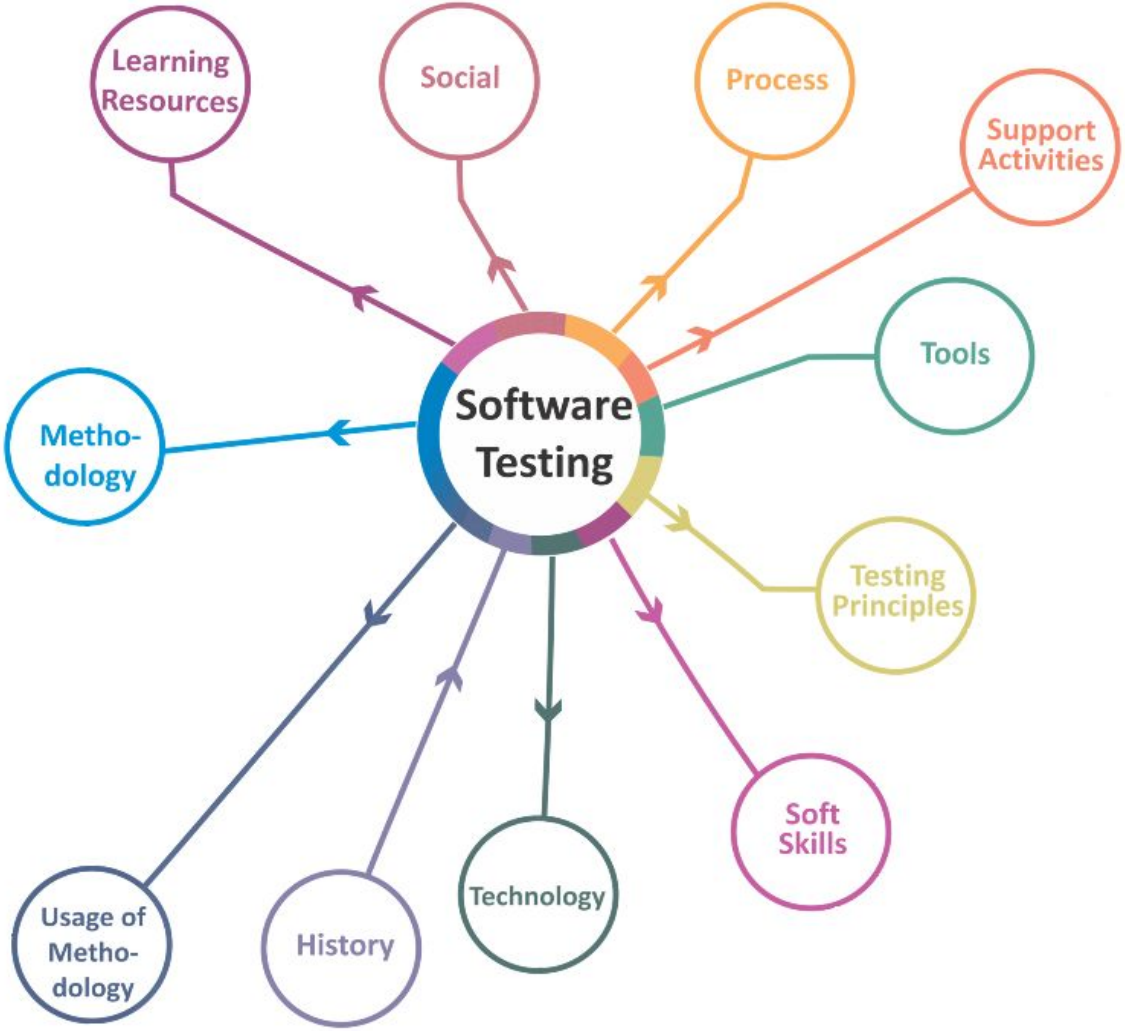
Build Verification Test.

Если сборка не соответствует критериям качества, то команда тестирования вправе ее отклонить (Reject), приложив список ошибок.

Дальнейшие варианты действий:

- Если сборка не работает по вине билд - мастера, то принимается решение о проведении перевыкладки версии (Re - deploy);
- Если сборка действительно не соответствует критериям качества, то производится откат (Roll back) на предыдущую версию.


Software
Testing


Learning Resources
Social
Process
Support Activities
Tools
Testing Principles
Soft Skills
Technology
History
Usage of Metho-dology
Metho-dology
Software Testing


Areas of Software testing
Economics
Boehm
Software crisis
Proof of correctness
Formal proof
Model based testing
Test tools
Test automation
Selenium
Fitnesse
soapUI
Coverage
Test techniques
Risk based testing
Specification based testing
Quality characteristics
Usability
Performance
Security
Localization
Principles
Exhaustive testing is impossible
Testing shows the presence of bugs
Defect clustering
Pesticide paradocs
Community
Conferences
Courses
Magazines
Testing organization
Test centre of excellence
Outsourcing
Crowdsourcing
Testing processes
Test process improvement
Models of testing
Debugging
Verification & validation
Standards
V-model, W-model
Testing quadrant
Test types
Unit testing
System testing
Acceptance testing
Regression testing
Chain testing
Roles of testing
Quality assurance
Quality director
Quality advocate
Quality engineer
Agile tester
Disruptive agent of change

Вопросы и ответы

https://www.guru99.com/smoke-sanity-testing.html

https://software-testing.org/testing/otlichie-sanitarnogo-sanity-testing-ot-dymovogo-smoke-testing-vidov-testirovaniya.html