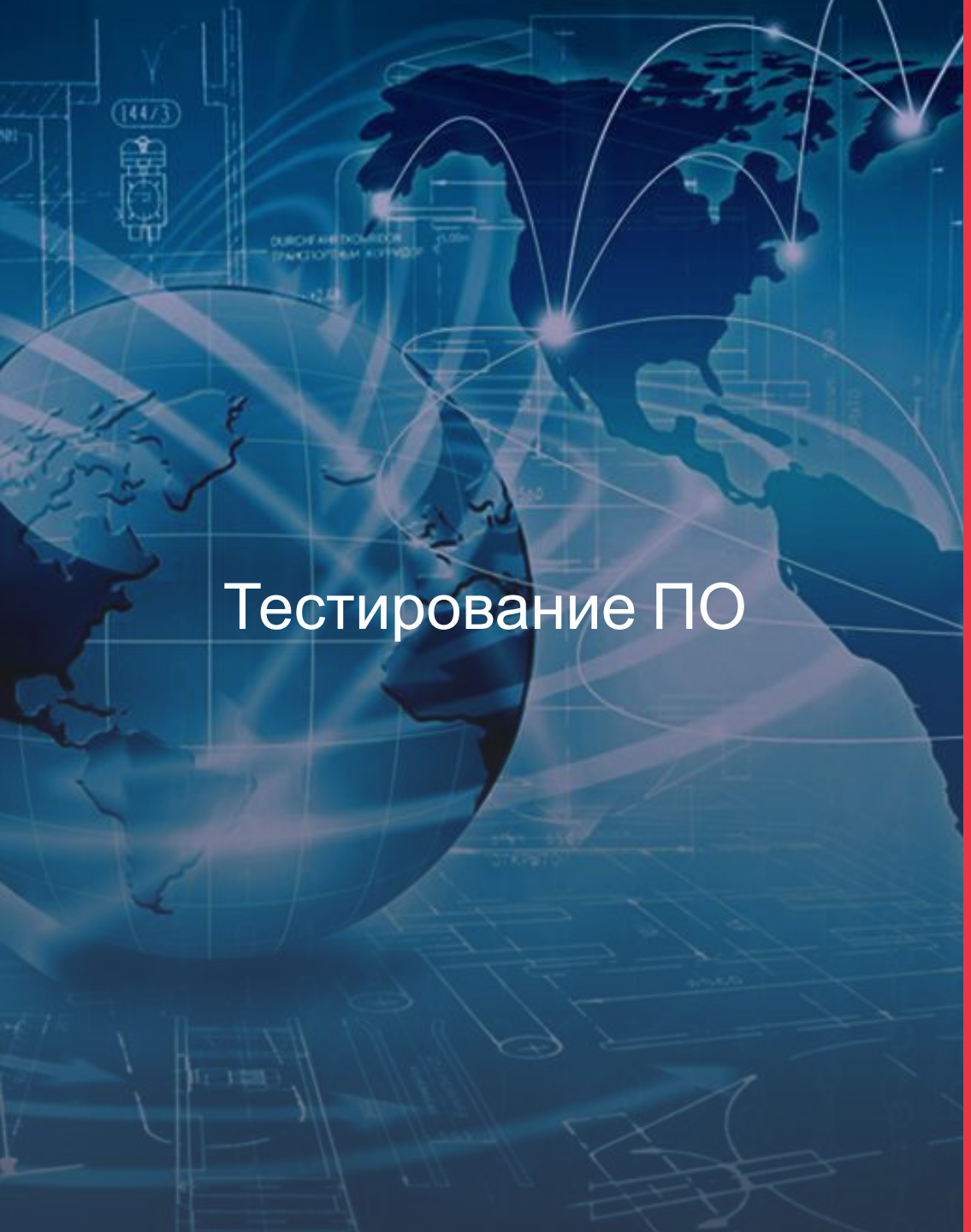


Тестирование ПО

Education
Academy

WWW.ITEDUCATE.COM.UA

Семенюк Марина

- Инструктор **IT Education Academy**

КОНТАКТНЫЕ ДАННЫЕ

mersik@gmail.com

IT Education Academy

Виды тестирования по доступности к исходному коду

Стеклянный/Белый ящик (Glass/White Box) – этот метод основан на том, что разработчик теста имеет доступ к исходному коду программ и может писать код, который связан с библиотеками тестируемого ПО.

Черный ящик (Black Box) – этот метод основан на том, что тестировщик имеет доступ к ПО только через те же интерфейсы, что и заказчик или пользователь, либо внешние интерфейсы, позволяющие другому компьютеру либо другому процессу подключиться к системе для тестирования

Серый ящик (Grey Box) - сочетает элементы двух предыдущих подходов.

Виды тестирования по выполнению программного кода

Статическое тестирование производится без запуска программного кода продукта.

Тестирование осуществляется путем анализа программного кода (code review) или скомпилированного кода. Анализ может производиться как вручную, так и с помощью специальных инструментальных средств. Целью анализа является раннее выявление ошибок и потенциальных проблем в продукте.

С помощью code review на раннем этапе могут быть выявлены ошибки в коде продукта. Как правило code review производится самими разработчиками.

Примерами ошибок, которые потенциально можно выявить с помощью автоматического статического тестирования, могут быть:

- утечки ресурсов (утечки памяти, неосвобождаемые файловые дескрипторы и т.д.)
- возможность переполнения буфера (buffer overflows)
- ситуации частичной (неполной) обработки ошибок.

Как правило, результатом автоматического анализа кода является список рекомендаций для ручного review некоторых участков кода, потенциально содержащих ошибки.

При **динамическом тестировании** необходим запуск кода приложения.

Класс динамического тестирования делится на множество типов, каждый из которых зависит от:

- Доступа к коду (тестирование черным, белым и серым ящиками).
- Охвата тестируемой программы (модульное, системное, интеграционное тестирование и другие).
- Области использования приложения (функциональное, нагрузочное, тестирование безопасности и пр.).

Динамическое тестирование помогает определить свойства программы по результатам ее выполнения – соответствие функциональным требованиям, работа в разных средах, разных версий, локализация, нагрузка, работа с входными и выходными данными, – типов динамического тестирования очень много.

Динамическое тестирование позволяет проверить созданный или создаваемый продукт на соответствие требованиям с точки зрения функциональности, эффективности работы и т.д.

Дымовое тестирование (Smoke Testing)

Понятие **дымовое тестирование (smoke testing)** пошло из инженерной среды:

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

В области же тестирования программного обеспечения, оно применяется для поверхностной проверки всех модулей приложения на предмет работоспособности и наличия быстро находимых критических и блокирующих дефектов.

В случае если **“smoke test” – failed**, вы отправляете приложение на доработку.

Санитарное тестирование (Sanity testing)

Санитарное тестирование - это узконаправленное тестирование достаточное для доказательства того, что конкретная функция работает согласно заявленным в спецификации требованиям.

Является подмножеством регрессионного тестирования.

Используется для определения работоспособности определенной части приложения после изменений произведенных в ней или окружающей среде. Обычно выполняется вручную.

Отличие санитарного тестирования от дымового (Sanity vs Smoke testing)

В некоторых источниках ошибочно полагают, что санитарное и дымовое тестирование - это одно и тоже.

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.

Регрессионное тестирование (Regression testing)

Регрессионное тестирование - это вид тестирования направленный на проверку изменений, сделанных в приложении или окружающей среде (починка дефекта, слияние кода, миграция на другую операционную систему, базу данных, web 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.

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.

Сам по себе термин "Регрессионное тестирование", в зависимости от контекста использования может иметь разный смысл.

Сэм Канер, к примеру, описал 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

Ad-Hoc тестирование
(интуитивное тестирование)

Цель Ad-Hoc testing определить, что **может поломаться**, когда пользователь делает что-то **неправильно**. Это тестирование без правил.

Пример.
Необходимо протестировать поле для ввода даты, применяя ad-Hoc тестирование, можно ввести в поле что-то наподобие 01/*1/2015.

Это импровизационное **негативное** тестирование.
В любой момент процесса тестирования, можно задать себе вопрос: а что будет, если…

Важной особенностью интуитивного тестирования является то, что оно проводится без тест-кейсов и спецификации. Пользователь о приложении не знает ничего.

Преимущества = недостатки
Случайность
Иногда невозможность повторить баг
Непредсказуемые варианты исходов
Нетребовательность к тестировщику
Не нужно время для подготовки

Exploratory Testing
(исследовательское тестирование)

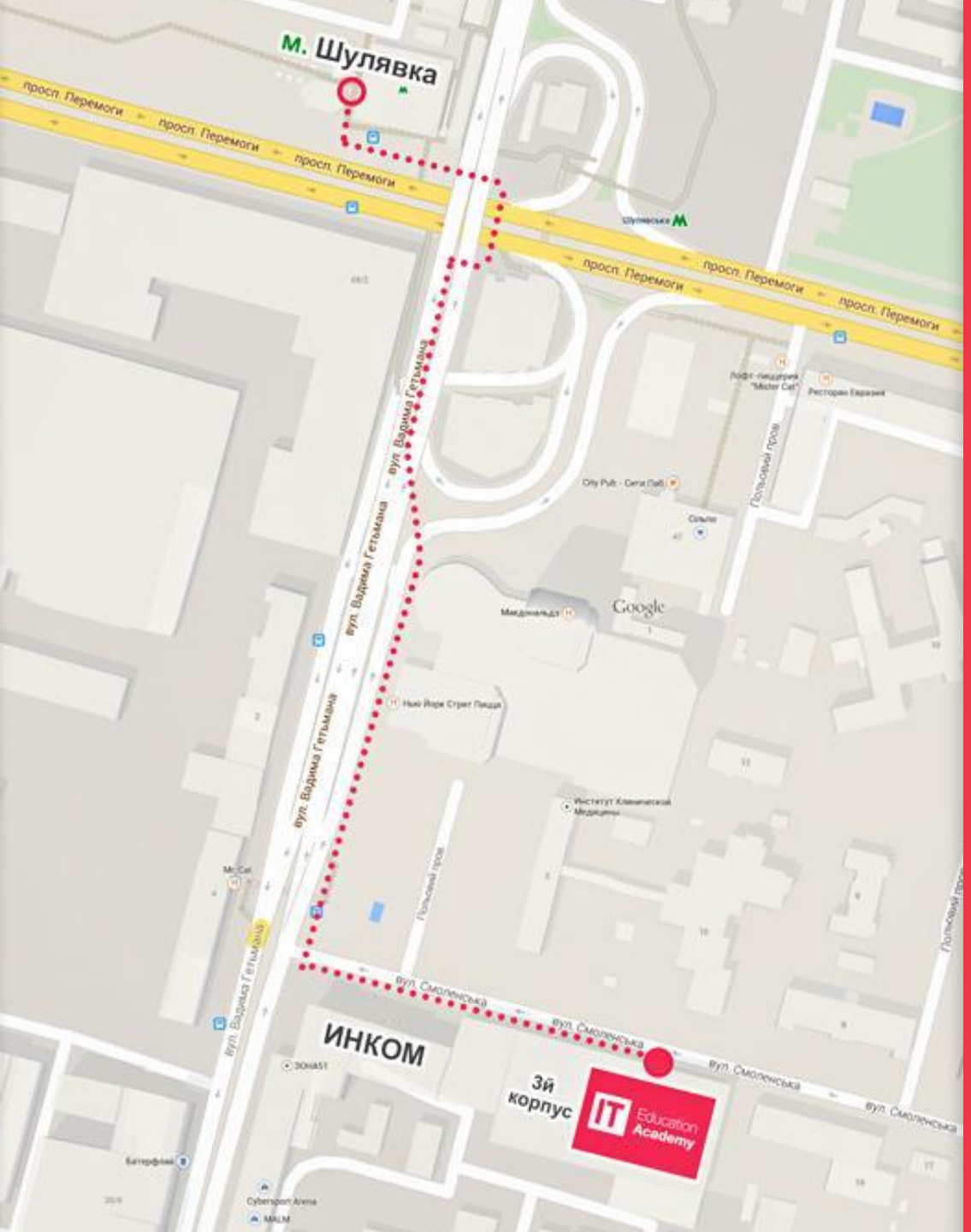
тип тестирования, выполняемый тестировщиком вручную без определенного заранее сценария (Test Case) и направленный на поиск ошибок в функциональности ПО путем попыток создания нестандартных ситуаций, при которых программа может давать сбои.

Для грамотного исследовательского тестирования у тестировщика должно быть общее представление о продукте или функционале.

Следующий шаг чаще всего определяется результатом выполнения предыдущего шага. Разработка и выполнение тест-кейсов происходит одновременно. По мере тестирования составляется пользовательский сценарий. Тесты придумываются на лету.

Преимущества = недостатки

Минимум подготовки
Более интеллектуальная работа
Построение тестовых сценариев “на лету”



IT Education Academy

КОНТАКТНЫЕ ДАННЫЕ

ITEA
ул. Смоленская, 31-33, корп.3
Киев
03005

+38 044 590 08 38
facebook.com/Iteducate
info@iteducate.com.ua
iteducate.com.ua