Программа курса “Введение в тестирование ПО” (20 часов)

Октябрь - Ноябрь, 2017

Тестирование мобильных приложений

Учитывая, что современные репрезентативные смартфоны стоят недешево, очень заманчиво использовать бесплатные и простые способы тестировать мобильные версии сайтов – например, браузерные симуляторы – вместо устройств как таковых.

Тестирование мобильных приложений

- **User Agent**

Достоинство этого инструмента – это возможность быстро протестировать текстовые поля и, тестируя локализацию, найти непереведенные участки текста и места, где текст вообще не отображается. Однако он только имитирует отображение для определенной платформы, и неверно масштабирует дисплей. В результате проверить, к примеру, длинные названия полей на немецком и убедиться, что они не ломают верстку или расположение кнопок, довольно сложно.

Тестирование мобильных приложений

Инструменты разработчика

Преимущества:

Адаптивные инструменты позволяют быстро протестировать верстку.

Легко получить доступ к разнообразным эффективным инструментам.

Недостатки:

Инструменты разработчика отличаются от браузера к браузеру, что приводит к тому, что тестирования в одном браузере недостаточно для целей вашего приложения.

Нельзя доверять функциональным тестам.

Тестирование мобильных приложений

Симулятор (модель оригинального ПО, в которой реализуется логика работы этого ПО (частично или полностью), имитируется поведение ПО, копируется его интерфейс.**)**

Преимущества:

Позволяют провести большую часть функционального тестирования – ограничения описаны в разделе недостатков ниже.

Можно использовать для smoke-тестирования

Улучшают тестовое покрытие, позволяя тестировать на конфигурациях железа, недоступных внутри компании.

Недостатки:

Зачастую требуют установки целой экосистемы, что может занять много времени: IDE, подписки и учетки для iOS-платформы, одна версия SDK для каждой версии OS, нуждающейся в тестировании.

Невозможно оценить пользовательский опыт: отклик на тап, проблемы "толстых пальцев", соответствие расположения элементов друг другу в процессе использования, и т. д.

Полагаются на ресурсы компьютера, невозможно оценить реальную производительность.

Не дают доступа к всем функциям ОС/устройства/браузера (например, к камере).

Тестирование мобильных приложений

- **Эмулятор (**полнофункциональный аналог оригинального ПО, либо его версия, в которой может быть предусмотрен ряд ограничений по функционалу, возможностям и поведению ПО.**)**

Преимущества:
эмуляторы предположительно должны работать на схожей с устройством конфигурации оборудования, устраняя недостаток, упомянутый выше для симуляторов
Недостатки:
Полагаются на ресурсы компьютера, невозможно оценить реальную производительность.
Не дают доступа к всем функциям ОС/устройства/браузера (например, к камере).

Тестирование мобильных приложений

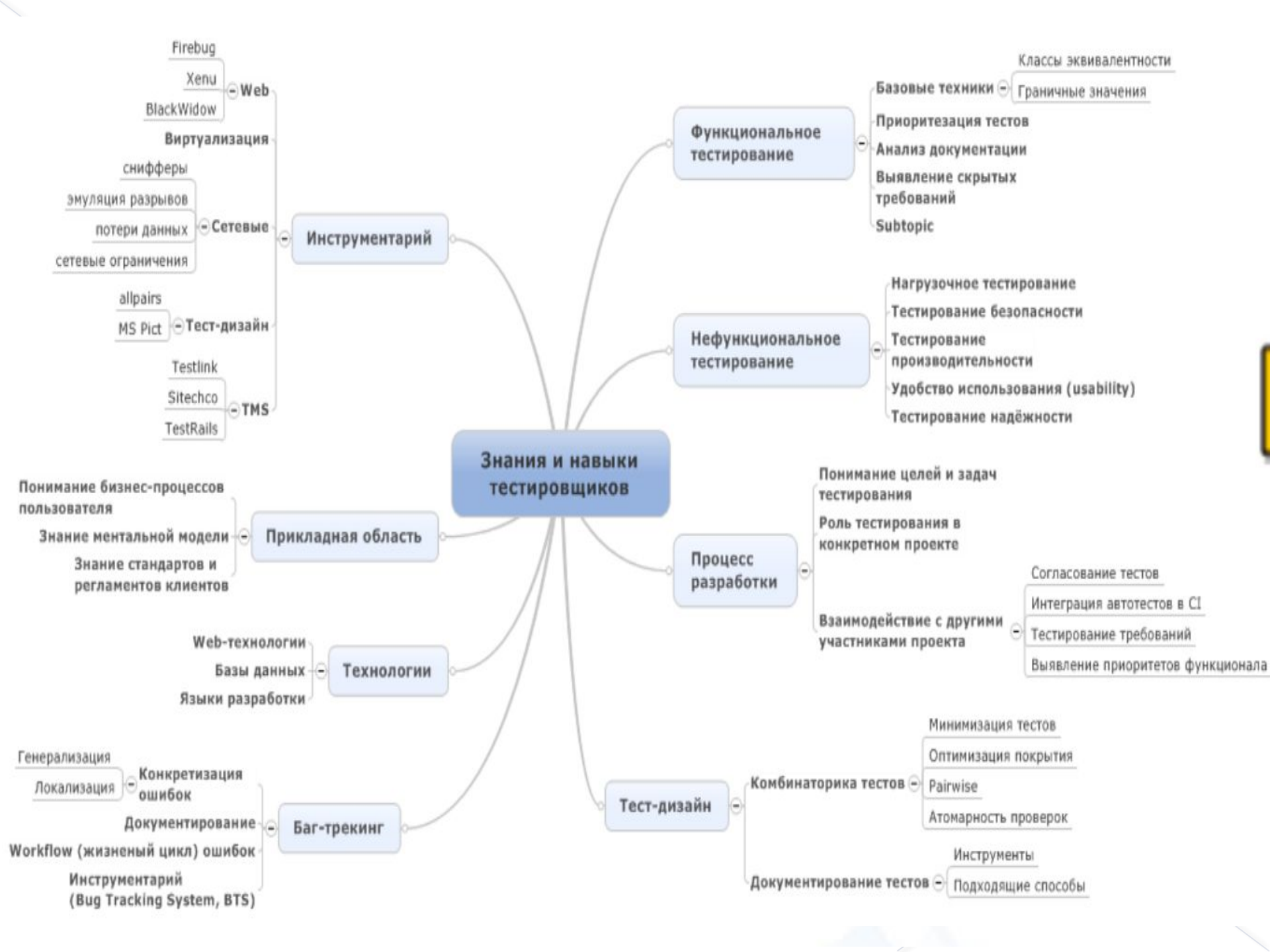
- **Физическое устройство**

Преимущества:
Наилучший инструмент для тестов, опирающийся на реальные ресурсы железа и ПО.
Позволяет оценить качество изображений.
Недостатки:
Стоимость устройств, которую необходимо закладывать в бюджет.
Необходимость делиться устройствами с другими членами команды может превратиться в "бутылочное горло" проекта: распланируйте управление устройствами внутри команды заранее!

Знания и навыки тестировщиков

- Что нужно знать и уметь хорошему тестировщику?
- Как эти знания получить?
- Как определить, каких знаний вам не хватает?

В этой презентации структурированы знания, необходимые
активным развивающимся тестировщикам, и собраны ссылки на
полезную литературу и тренинги по рассматриваемым темам.


Знания и навыки тестировщиков
Инструментарий
Web
Firebug
Xenu
BlackWidow
Виртуализация
Сетевые
снифферы
эмуляция разрывов
потери данных
сетевые ограничения
Тест-дизайн
allpairs
MS Pict
TMS
Testlink
Sitechco
TestRails
Прикладная область
Понимание бизнес-процессов пользователя
Знание ментальной модели
Знание стандартов и регламентов клиентов
Технологии
Web-технологии
Базы данных
Языки разработки
Баг-трекинг
Конкретизация ошибок
Генерализация
Локализация
Документирование
Workflow (жизненый цикл) ошибок
Инструментарий (Bug Tracking System, BTS)
Функциональное тестирование
Базовые техники
Классы эквивалентности
Граничные значения
Приоритезация тестов
Анализ документации
Выявление скрытых требований
Subtopic
Нефункциональное тестирование
Нагрузочное тестирование
Тестирование безопасности
Тестирование производительности
Удобство использования (usability)
Тестирование надёжности
Процесс разработки
Понимание целей и задач тестирования
Роль тестирования в конкретном проекте
Взаимодействие с другими участниками проекта
Согласование тестов
Интеграция автотестов в CI
Тестирование требований
Выявление приоритетов функционала
Тест-дизайн
Комбинаторика тестов
Минимизация тестов
Оптимизация покрытия
Pairwise
Атомарность проверок
Документирование тестов
Инструменты
Подходящие способы

1. Функциональное тестирование

Функциональное тестирование – проверка, что продукт успешно работает и выполняет то, что требуется пользователю (даже если требуемое ему не оговорено или пропущено в спецификациях продукта).

Знания и навыки	Зачем нужны
Базовые техники проектирования тестов (классы, границы)	Оптимизация тестов, исключение манки-тестинга, осознанность в выборе тестов
Техники исследовательского тестирования (чит-листы, тест-туры, мнемоника MUTII, шляпы пользователей и т.д.)	Генерация оптимальных тестов «на лету», исходя из текущего статуса продукта и знаний о его использовании
Приоритезация тестов (на основе приоритетов пользователя, проектных задач и рисков возникновения ошибки)	Нахождение дефектов в требуемой проекту последовательности, снижение вероятности пропуска критичных багов
Анализ требований (тестирование спецификаций, поиск нестыковок, выяснение недостающей информации по продукту)	Удовлетворение клиента, получение продукта требуемого качества, непропуск критичных ошибок

Книги:

✓Тестирование программного обеспечения (Сэм Канер, Джек Фолк, Енг Кек Нгуен)

✓Software Testing (Ron Patton)

✓The Art of Software Testing (G. Myers)

Тренинги:

✓ Школа Успешных Тестировщиков

✓ Курс практического тестирования для начинающих

✓ Базовый курс по тестированию

✓ Тестирование методом свободного поиска

Практика:

✓ Тестирование на сервисах Utest.com и Fixber.ru

✓ Общение с аналитиком и выяснение приоритетов

✓ Внедрение техник проектирования и исследовательского тестирования в ежедневную работу

2. Нефункциональное тестирование

Нефункциональное тестирование – проверка того, КАК работает продукт (насколько удобно, насколько быстро, в каких условиях и окружениях, с каким количеством одновременных пользователей и т.д.)

Стать «экспертом в нефункциональном тестировании» невозможно, т.к. каждая из областей содержит очень много необходимых навыков и знаний. В небольших объёмах они нужны любому тестировщику, но для глубокого изучения необходима специализация в отдельном виде нефункционального тестирования.

Знания и навыки	Зачем нужны
Методология нефункционального тестирования	У каждой области нефункционального тестирования есть своя методология: используемые техники, подходы, инструменты. Только базируясь на них можно оценить нефункциональные требования к ПО, затратив приемлемое количество усилий
Прикладная область	Для тестирования защищённости необходимо быть специалистом по безопасности, а для тестирования удобства использования нужно разбираться в проектировании интерфейсов, usability и UX (user experience). Каждая область требует свой набор знаний, необходимых для её тестирования.

Книги:

✓How to break web software, James Whittaker (Безопасность)

✓How to break software security, James Whittaker (Безопасность десктоп-приложений)

✓Проектирование веб-интерфейсов Б.Скотт, Т.Нейл (Удобство использования)

Тренинги:

✓ Тестирование защищённости веб-приложений

✓ Тестирование производительности

✓ Школа Успешных Тестировщиков

Практика:

✓ Использование нефункционального тестирования в своём проекте

✓ При выборе специализации на каком-либо виде тестирования – его проведение на различных, в том числе сторонних, сервисах

✓ Выбор подходящей компании для трудоустройства (в СНГ не так много фирм, в которых налажено нефункциональное тестирование).

3. Баг-трекинг

Баг-трекинг– процесс работы над найденными ошибками (их уточнение, регистрация в баг-трекере, предоставление требуемой информации, проверка исправлений).

Баг-трекинг является интерфейсом взаимодействия тестирования и других участников проекта, поэтому тестировщиков зачастую оценивают именно по качеству работы с дефектами.

Знания и навыки	Зачем нужны
Генерализация и локализация ошибок	Чтобы точно указать разработчикам, в чём именно ошибка, и сократить время на её исправление и последующую перепроверку
Описание дефекта	Чтобы все участники проекта могли понять смысл дефектов, затратив минимум усилий и времени (когда дефект регистрируете не вы сами, понять со стороны его значительно сложнее, чем кажется!)
Жизненный цикл дефекта, его Workflow	Жизненный цикл в каждой компании разный и создаётся исходя из команды, процессов, продукта. Следование процессу (и его улучшение) позволяют экономить ресурсы, время, и повышать качество работы над ошибками

Книги:

✓Тестирование Дот Ком (Роман Савин)

Тренинги:

✓ Школа Успешных Тестировщиков

✓ Курс практического тестирования для начинающих

Практика:

✓ Тестирование на сервисах Utest.com и Fixber.ru

✓ Участие в бета-кампаниях Software-Testing.Ru

✓ Спросить у разработчиков, что им нравится в заведении дефектов, а что стоит улучшить

✓ Просмотр публичных баг-трекеров и регистрация ошибок по известным сервисам, к примеру:

- ✓ Mozilla Firefox – Bugzilla
- ✓ Mantis
- ✓ Chromium BTS
- ✓ Adobe FlashPlayer BTS

4. Процесс разработки

Процесс разработки – набор договорённостей и правил, по которым работает проект. В их число входят принятые документы, шаблоны, стандарты, процедуры, роли, последовательности выполнения задач и т.д.

В каждом типе процесса от тестирования требуются различные задачи: в гибких (agile) методологиях это помощь в выпуске, в формальных моделях это следование процессу и контроль качества.

Знания и навыки	Зачем нужны
Процесс, используемый в вашем проекте	Если процесс нигде не описан (а так обычно и бывает), это не значит, что его нет. Все участники команды что-то ожидают друг от друга и от проекта в целом, и общее понимание процесса необходимо для совместной продуктивной работы.
Общая информация о проектных методологиях	В каждой проектной методологии есть свои особенности, влияющие на всех участников, в том числе и на тестировщиков. Зачастую наиболее успешные тестировщики – те, которые смогли подстроиться под используемые в проекте процессы и принести, исходя из них, максимум пользы.

Книги:

- ✓Дедлайн, Том Демарко
- ✓Гибкое тестирование, Лайза Криспин, Джанет Грегори

Тренинги:

- ✓ Базовый курс по управлению тестированием
- ✓ Тестирование в стартапах

Практика:

- ✓ Общение с участниками проекта, техника «5 почему»
- ✓ Изучение конкретных проектных методологий:
  - ✓ RUP
  - ✓ Scrum
  - ✓ XP

Дополнительные ссылки:

- ✓ Тестирование в RUP
- ✓ Доклад Асхата Уразбаева о тестирование в Agile

5. Тест-дизайн

Тест-дизайн, или проектирование тестов – это определение «что должно тестироваться». У нас никогда не бывает достаточно ресурсов на полное тестирование, поэтому мы лишь составляем список того, что мы успеем протестировать за отведённое время.

Тест-дизайн – пожалуй, самая важная и самая необходимая область для развития в тестировании, которая при этом является и самой обширной.

Знания и навыки	Зачем нужны
Анализ проектной спецификации	Только умея читать документы между строк, анализировать взаимосвязи, выявлять пропущенную информацию и задавать правильные вопросы, тестировщики могут проверять именно те моменты, которые необходимо тестировать.
Проектирование и комбинаторика тестов	Вне зависимости от того, документируете вы тесты или нет, в любом случае вам необходимо их детальное, вдумчивое проектирование. За счёт оптимальной комбинаторики тестов возможно сокращение затрат и повышение тестового покрытия одновременно. В качестве примеров техник проектирования тестов можно назвать pairwise (попарное тестирование), state & transition testing (тестирование состояний и переходов), decision tables (таблицы решений) и т.д.
Документирование тестов	Некоторые проекты губит отсутствие задокументированных тестов – а некоторые губит их избыточность и бюрократия. Необходимо уметь выбирать правильный способ и объём документирования, так же как и формат: чек-листы или тест-кейсы, детальные или поверхностные, в таблицах или специализированных тест-менеджмент системах (TMS).

Книги:

✓Guide to Software Test-Design, Lee Copeland

✓Essential Test Design, Torbjorn Ryber

Тренинги:

✓ Планирование тестирования и проектирование тестов

✓ Практикум по тест-дизайну

Практика:

✓ Максимальный отказ от хаотического тестирования «по наитию» в сторону продуманного тест-дизайна

6. Прикладные навыки

Помимо специфичных для тестировщиков знаний нам в ежедневной работе необходимы также **знания, специфичные для нашего продукта.**

Продукты для бухгалтеров и фармацевтов, для корпоративных и домашних пользователей, системные, мобильные и веб-, однопользовательские и клиент-серверные, все эти типы продуктов необходимо тестировать по-разному

Знания и навыки	Подробнее
Прикладные знания	✓ Знания бухгалтерского учёта, стандартов и регламентов для тестирования бухгалтерского ПО ✓ Знание сетевых протоколов для тестирования сетевого ПО
Знания технологий	✓ Понимание веб-технологий при тестировании веб и мобильных систем при тестировании мобильных приложений ✓ Умение писать SQL запросы при тестировании БД и навыки системного администрирования при тестировании серверного ПО ✓ Какие инструменты помогут вам сэкономить время: proxy-сервера, анализаторы кода, снифферы, скриншотеры, и т.д.
Понимание пользователя	✓ Стандартные способы использования вашего продукта, регулярность использования того или иного функционала ✓ Стандартные окружения, браузеры, операционные системы и другие условия использования вашего продукта ✓ Квалификация ваших пользователей: сисадмины или домохозяйки? ✓ Какие задачи решает ваш пользователь? Каким образом? Как часто?

Тренинги:

✓ Тестирование веб-приложений
✓ SQL для тестировщиков
✓ Тестирование веб-сервисов

На данный момент существует совсем немного специализированных тренингов по тестированию с учётом технологий, используемых при разработке ПО.

Значительно важнее регулярно общаться с аналитиками и разработчиками, читать профессиональную литературу по вашей прикладной области и анализировать пропущенные ошибки: зачастую именно они являются ключом и ответом на вопрос «каких знаний нам не хватило для обнаружения этой ошибки?»

В о п р о с ы ?