Управление Качеством ПО

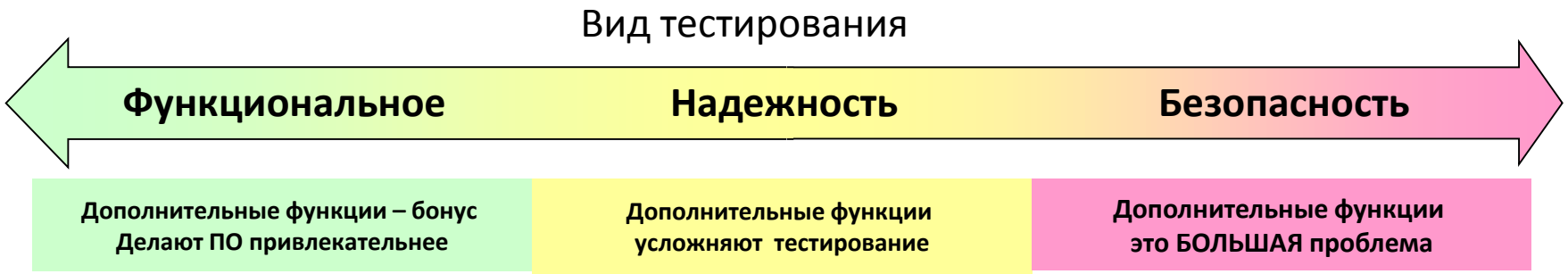
Лекция 3

Минеева Ольга

Тестирование безопасности

Разработано для курса НФ ВШЭ 4 курс бакалавриата. Факультет БИМП.
Авторы: Виценко А.Ю., Крюков А.К., Морёнов О.А, Пряхин И.В., Семенов Д.С., Чиликин Е.В.

В чем особенность тестирования на наличие уязвимостей



- Сходно: цель тестирования – уменьшить вероятности проявления ошибки
- Отличия:
  - При функциональном тестировании проверяется работоспособность заявленных функций ПО
  - Security - тестирование способности противостоять атакам

Симптомы позволяющие выявить уязвимость

- Crash!
- Нестабильность в работе приложения и/или системы в целом
- “Подвисание” программы
- Неожиданное увеличение объёма памяти используемой приложением
- Сообщения об ошибках
  - Внимательно читайте сообщения об ошибках. Даже если они не имеют никакого смысла это может сказать о многом
  - Позволяют лучше понять реальную структуру программы не всегда отраженную в документации
  - Отсутствие сообщения об ошибке может означать отсутствие проверки на их наличие

Memory corruption

- Memory Corruptions – одна из наиболее распространённых и легко используемых ошибок
  - Переполнение буфера
  - Выход за границы массива
  - Неправильное вычисление требуемого размера памяти
  - Неправильные спецификации формата (%s)
  - Искажение данных в результате не корректной сериализации в многопоточном коде

Test Cases для Memory Corruptions

- Ввод очень длинной строки
- Переполнение строк типа <name>=<value>
- Переполнение строк типа путь на файловой системе
  - Drive letter e.g. **<BO>:**\folder\file.txt
  - Folder name e.g. C:**<BO>**\file.txt
  - Filename e.g. C:\folder**<BO>** . **<BO>**
- Переполнение URL
- “Поиграться” размером или смещением при работе с массивами и строками:
  - Отрицательные значения
  - Нулевые значения
  - Значения выходящие за границу массива
  - Значения кратные или близкие к 2 в степени n
- Удалить/заменить разделители (‘\0’, ‘/’, ”, CRLF, ‘<‘, ‘>’)
- Использовать спецификатор формата в поле ввода. Например: %s%s%12s

Обход контроля доступа

- Разграничения доступа на основе имени защищаемого ресурса легко обойти
  - Альтернативные имена могут быть использованы для файлов, URL или устройств
  - Следующие пути идентичны в Windows:
    - C:\Qqq.Txt
    - C:\QqQ.txt
    - c:\qqq.txt::$DATA
  - Аналогично для URL:
    - http://www.qqq.com/secret
    - http://www%2eqqq%2ecom%2fsecret

Инструменты для статического анализа кода

- Наиболее известны – Klocwork, Coverity® Development Testing Platform и Intel® Inspector XE (*Static Analysis feature)*
- Позволяют обнаружить большинство типовых ошибок на этапе написания кода и избежать значительных затрат на тестирование в runtime
- Покрывают как часто так и редко используемые участки кода.
- Учитывают возможные последствия при анализе важности проблем
- Легко интегрируются в build системы

Важное замечание

- До начала любых работ связанных с тестированием ПО на наличие уязвимостей необходимо подготовить изолированное окружение.
- Не допустимо использовать “чужие” системы
- Неправомочный доступ к системе может повлечь за собой ответственность по статье 272 УК РФ – “**Неправомерный доступ к компьютерной информации**”

Заключение по тестированию безопасности

- Тестирование ПО на отсутствие уязвимостей в системе безопасности важный этап в общем цикле тестирования
- Используемые методы в значительной степени отличаются от других видов тестирования
- Даже если при разработке ПО не ставились специальные требования к безопасности применяемые подходы позволяют значительно повысить общую стабильность ПО
- Стоимость исправления ошибок в безопасности обнаруженных после выпуска выше чем для других выявленных ошибок

Основы автоматизации тестирования

Цели автоматизации

- Повышение производительности труда
- Минимизация человеческого фактора, повышение надежности
- Удешевление процессов (более эффективное использование ресурсов)
- Повышение скорости разработки

- Автоматизация тестирования – **инвестиции** ресурсов с целью экономии б***о***льшего числа ресурсов в дальнейшем

Возможности автоматизации (1)

- Повышение производительности труда за счет автономности
  - Старт автоматики по событию, «кнопке» и т.д.
  - Взаимодействие с продуктом/компонентой
  - Анализ результатов, сравнение с эталоном
  - Формирование читаемых очетов

- Минимизация человеческого фактора, повышение надежности:
  - Ниже вероятность ошибок (в следствие потери концентрации или отсутствия экспертизы)
  - Повторяемость
  - Стабильность (у автоматики нет отпусков или больничных)

- Удешевление процессов (эффективное использование ресурсов - 24/7)
  - Возможное сокращение тестовых машин

Возможности автоматизации (2)

- Повышение скорости работы ...
  - ... тестовой команды: автоматический тест в среднем занимает меньше времени, чем ручной
  - ... разработки в целом: тесты идут за ночь вместо нескольких дней
- Генерация данных:
  - Возможность увеличения тестового покрытия без больших инвестиций в разработку автоматизации
- Сохранение логов:
  - Отслеживание прогресса (% выполнено)
  - Показатель выполнения/невыполнения тестов
- Моделирование тестовых сценариев: smoke- и стресс-тестирование, тестирование устойчивости к атакам ...

Сравнение ручного и автоматизированного подходов:

Критерий	Ручное	Автоматическое
Задание входных значений	Гибкость в задании данных. Позволяет использовать разные значения на разных циклах запуска тестов, расширяя покрытие	Входные значения строго заданы
Проверка результата	Гибкая, позволяет тестировщику оценивать нечетко сформулированные критерии	Строгая. Нечетко сформулированные критерии могут быть проверены только путем сравнения с эталоном
Повторяемость	Низкая. Человеческий фактор и нечеткое определение данных приводят к неповторяемости тестирования	Высокая
Надежность	Низкая. Длительные *тестовые циклы* приводят к снижению внимания тестировщика	Высокая, не зависит от длины *тестового цикла*
Чувствительность к незначительным изменениям в продукте	Зависит от детальности описания процедуры. Обычно тестировщик в состоянии выполнить тест, если внешний вид продукта и текст сообщений несколько изменились	Высокая. Незначительные изменения в интерфейсе часто ведут к коррекции эталонов
Скорость выполнения	Низкая	Высокая
Возможность генерации тестов	Отсутствует. Низкая скорость выполнения обычно не позволяет исполнить сгенерированный набор тестов	Поддерживается

Тестирование интерфейса командной строки

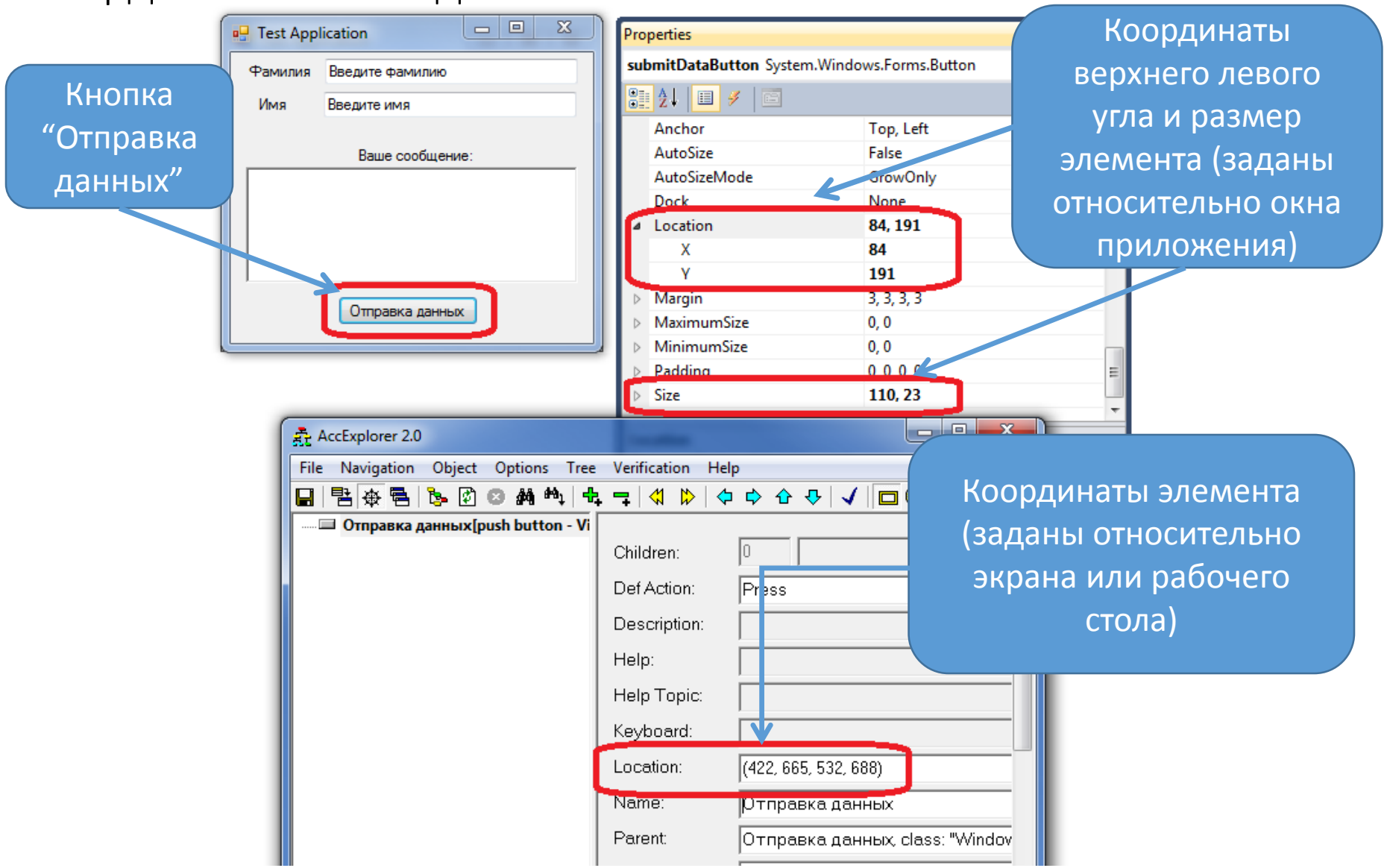
- При решении задачи автоматизации приложений с интерфейсом командной строки чаще всего решаются следующие задачи:
  - Запуск приложения с определенными аргументами
  - Получение результатов
  - Анализ результатов (анализ содержимого STDOUT, STDERR, полученных файлов и т.д.)
  - Вынесение вердикта о результатах – PASS или FAIL

Тестирование графического интерфейса пользователя

Рассмотрим основные подходы к автоматизации тестирования графического интерфейса пользователя:

- Координатный метод
- Распознавание образов
- Подход, использующий механизмы реализации специальных возможностей (accessibility) и особенности реализации некоторых GUI фреймворков

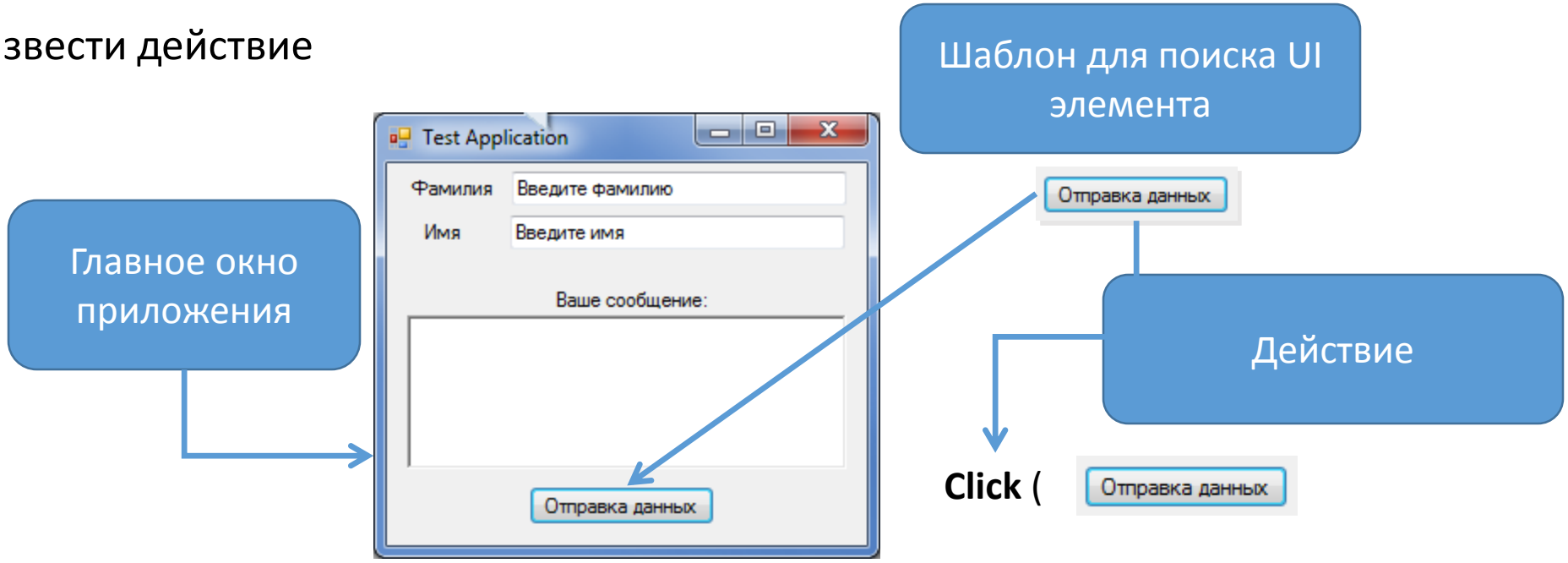
Тестирование графического интерфейса пользователя – координатный метод



Тестирование графического интерфейса пользователя – распознавание образов

Пример теста, построенного на системе распознавания образов:

- Запустить приложение
- Найти главное окно приложения
- Найти требуемый элемент, сравнив с шаблоном
- Произвести действие



Тестирование WEB-приложений

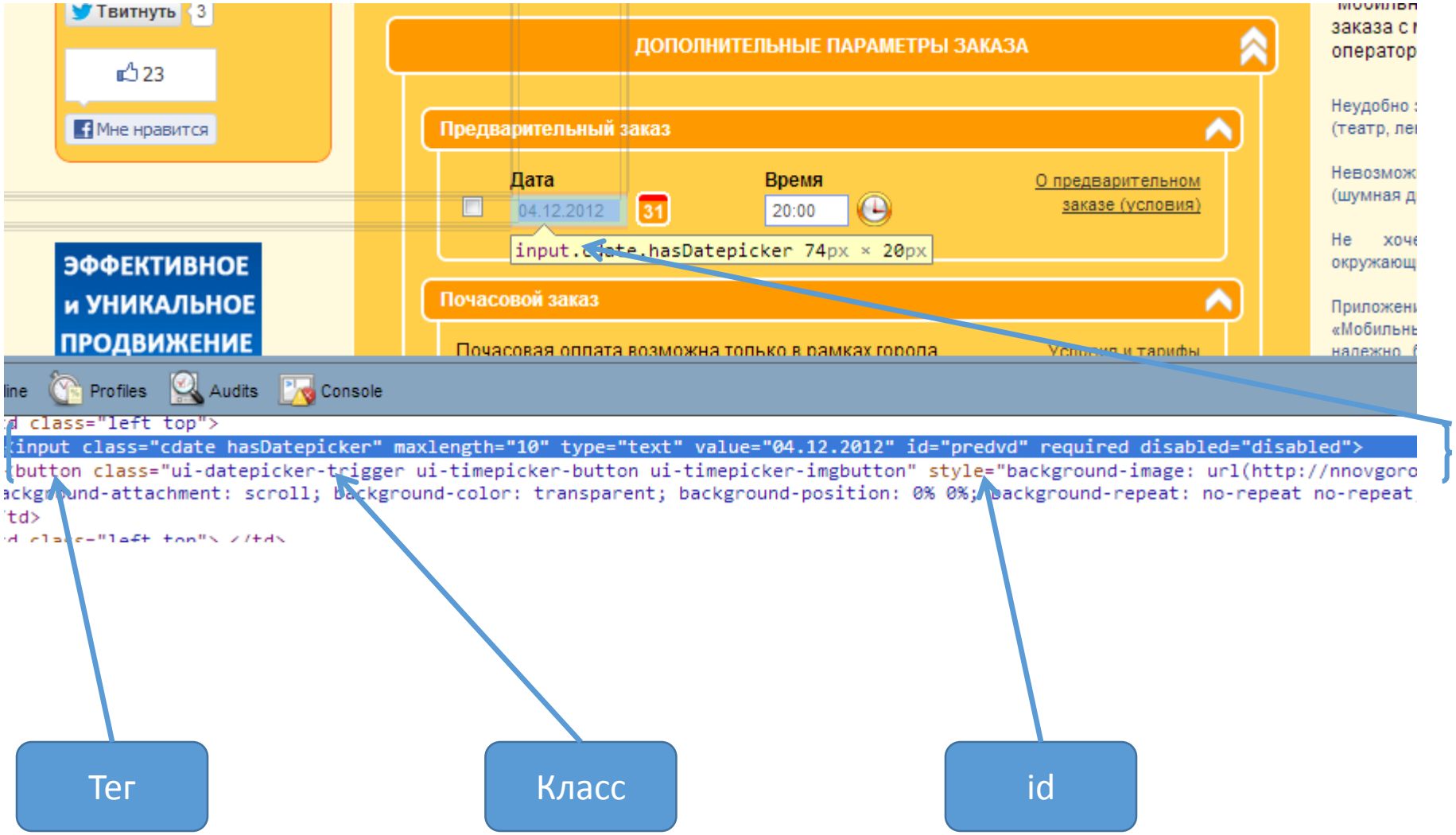
- Особенности тестирования Web-приложений:
  - В отличие от обычного графического пользовательского интерфейса Web-интерфейс отображается не самим приложением, а посредником – Web-браузером. Тестирование должно покрывать все поддерживаемые приложением версии браузеров
  - Чаще всего Web-приложения имеют клиент-серверную архитектуру (функциональность реализуется как на стороне сервера, так и на стороне клиента)
  - Описание пользовательского интерфейса предоставляется браузеру в стандартном представлении, в роли которого обычно выступает HTML
  - Представление данных отделено от логики приложения

Доступ к элементам пользовательского интерфейса в Web-приложениях

Поиск элементов графического интерфейса в Web-приложениях может выполняться следующими способами:

- По уникальному id
- По имени (name)
- По имени класса
- По имени HTML тега
- С использованием XPath (XML Path Language)
- С использованием CSS Selector’ов

Доступ к элементам пользовательского интерфейса в Web-приложениях (пример)



Проблемы автоматизации Web-приложений

- При автоматизации тестирования web-приложений возможны следующие проблемы:
  - Аттрибуты элемента типа ID (name) изменяются динамически
  - Несколько UI элементов имеют одинаковые id и (или) name
  - HTML5: доступ к элементам типа canvas и svg
  - Доступ к динамическому содержимому – Flash, Silverlight, Java Applets
  - Доступ к контенту, который содается «на лету». Например, с использованием JavaScript

Инструменты автоматизации – ч.1

Модульное тестирование:

- xUnit семейство (PHPUnit, NUnit, CPPUnit, и т.д.)

Тестирование графического интерфейса:

- Window Tester Pro – тестирование RCP и SWT приложений
- LDTP (Linux Desktop Testing Project) – тестирование GUI на Linux и (уже) Windows
- PyWinAuto – модуль для Python, предназначенный для тестирования GUI на Windows
- Sikuli – средство автоматизации GUI приложений, использующее распознавание образов

Тестирование Web-приложений:

- Selenium
- Tellurium
- Watir

Инструменты автоматизации – ч.2

Тестирование на соответствие стандартам (conformance testing):

- Khronos Conformance (соответствие стандартам OpenCL, OpenGL, WebGL и т.д.)

Тестирование поддержки специальных возможностей (accessibility testing):

- UI Automation Verify (UIA Verify)
- Inspect

Тестирование безопасности:

- Microsoft SDL RegEx Fuzzer – средство для Fuzz тестирования
- Microsoft Attack Surface Analyzer – средство для сравнивания состояний системы

Оценка качества тестирования

Критерии готовности функциональности

- Специфицированы и утверждены критерии приемки
- Готовы сценарии тестирования, демонстрирующие достижение критериев приемки
- Готов код, предназначенный для тестирования
- Тесты и код зарегистрированы в системе контроля версий
- Программа из поставленного кода успешно компилируется и собирается на сервере сборки
- Приемочные тесты (unit, component, system) для построенной программы успешно выполнены
- Все остальные тесты были выполнены
- Пользовательская документация готова и обновлена
- Ответственный за продукт принял результат изменений

Критерии готовности продукта к выпуску

- Весь значимый функционал включен в релиз-кандидат
- Инспекция безопасности проведена успешно
- Тестовая команда уверена, что ни одна из добавленных функций не имеет значимого риска появления проблем на рабочей (продуктовой) среде, выполнены следующие критерии:
  - все обнаруженные ошибки высокой и очень высокой критичности исправлены
  - проведено конфигурационное тестирование
  - проведено минимальное тестирование локализованной версии (в разрезе стран, языков)
  - проверены специальные возможности (accessibility)
  - проверена функциональность с точки зрения пользователей (user experience)
  - проверена производительность
  - достигнуто соответствие бизнес-целям
- Каждая включенная функция была продемонстрирована и принята ответственным за продукт

Финальный отчет о результатах тестирования

- Наименование продукта
- Дата составления отчета
- Ссылки на спецификации, тестовые планы
- Список открытых дефектов
- Отчеты о результатах тестирования на всех уровнях (модульное тестирование, интеграционное тестирование, системное тестирование и т.д.)
- Информация об отклонениях от спецификаций и тестовых планов
- Оценка качества тестов
- Оценка качества продукта

Метрики процесса

- Дефекты (качество и количество)
- Тестовые сценарии
- Тестовое покрытие
- Скорость тестирования
- Эффективность планирования
- Взаимодействие с проектной командой

Метрики процесса: Дефекты

- Количество найденных дефектов
  - Общее количество
  - Распределение дефектов по важности (Bugs by Severity)
  - Распределение дефектов по приоритету (Bugs by Priority)
  - Отношение количества открытых дефектов к закрытым (% Open/Closed Bugs)
  - Количество дефектов на 1000 строк кода (Bugs/KLOC)
- Качество дефектов
  - Отношение количества повторно открытых дефектов к закрытым (% Reopened/Closed Bugs)
  - Отношение количества отклоненных дефектов к открытым (% Rejected/Opened Bugs)
  - Оценки экспертов и разработчиков

Метрики процесса: Тестовые сценарии

- Количество тестовых сценариев (# of Test Cases)
- Количества выполненных/успешно прошедших/непрошедших/ заблокированных тестовых сценариев (# of Test Cases Executed/Passed/Failed/Blocked)
- Количество невыполненных тестовых сценариев предназначенных к исполнению (# of Not Run Test Cases)
- Количество проведенных/завершенных/ непрошедших тестовых сессий (Total Test Sessions/Passes/Failures)

Метрики процесса: Тестовое покрытие

- Покрытие требований (Requirements Coverage)
- Покрытие кода (Code Coverage)
- Покрытие пользовательских сценариев
- Покрытие окружений (ОС, браузеры, hardware)
- Покрытие за цикл тестирования и покрытие на одной сборке

Метрики процесса: Скорость тестирования

- Скорость тестирования одной сборки
- Скорость полного тестового цикла
- Скорость приемочного тестирования
  - Как быстро проводится проверка валидности сборки с помощью приемочных тестов?
- Скорость заведения дефектов (Time to BTS)
  - Насколько быстро заводятся критичные дефекты?
- Эффективность планирования
  - Затраты факт / затраты план
  - Сроки факт / сроки план

Домашнее задание:

- По существующим тестовым планам провести тестовую сессию.
- Оценить проведенное тестирование: какие проблемы были выявлены, какие инструменты были использованы, сколько времени потребовалось, другое..
- Написать отчет в свободной форме