ИТМО ВТ

Тестирование ПО

Клименков С.В.
2013-2014 уч. Год
v.1.05 от 24.04.2015

ИТМО ВТ

Литература

- Иан Соммервилл. Инженерия программного обеспечения
- Рекс Блэк «Advanced Software Testing»,
- Рекс Блэк Ключевые процессы тестирования.
- ISTQB - International Software Testing Qualifications Board
  - www.istqb.org
  - www.rstqb.org
- P.Ammann, J.Offutt
  Introduction to Software Testing
  - www.cs.gmu.edu/~offutt/softwaretest/

ИТМО ВТ

Основы тестирования

1

ИТМО ВТ

Системы с программным обеспечением

- Бизнес-системы
- Аппаратура с ПО
- Потребительские товары
- Военные, космические системы
- Информационно-управляющие системы
- ...

Сильное давление со стороны бизнеса и гибких методологий

ИТМО ВТ

Термины

Люди ошибаются

- Mistake (Error) - Ошибка, просчет. (человека)
- Fault - Дефект, изъян. (ПО в результате ошибки)
- Failure – Неисправность, отказ, сбой. (Внешнее проявление дефекта)
- Error - Невозможность выполнить задачу вследствие отказа

Отказ м.б. следствием окружающей среды

ИТМО ВТ

Пример программы

```
public  int countPositive (int [ ] data) {
   int count = 0;
   for (int i = 1; i < data.length; i++) {
      if (data [ i ] > 0)
         count++;
   }
   return count;
}
```

Сколько здесь дефектов
и к чему они могут привести?

ИТМО ВТ

BUG

- Используется неформально
- Может обозначать
  - Дефект (fault)
  - Отказ (failure)
  - Невозможностью выполнить задачу (error)
  - Что то другое или ничего не обозначать

ИТМО ВТ

Примеры ошибок в отрасли

ИТМО ВТ

Уровни восприятия тестирования

- Уровень 0 – тестирование == отладка
  - Не отличает некорректное поведение и ошибки в программе
  - Не учитывает требования надежности и безопасности
- Уровень 1 - предназначение – показать корректность ПО

HW engineer

  - Невозможно доказать
  - Что значит “ошибок нет”?
  - Нет формальных правил

“Тестирование может показать наличие дефектов, а не их отстутствие”
Edgar Dijkstra

Уровни восприятия тестирования

- Уровень 2 - Демонстрация ошибок
  - Конфликт разработчиков и тестировщиков
- Уровень 3 - Тестирование может показать наличие ошибок
  - Используя ПО мы подвержены рискам
  - Риск – последствия незначительные
  - **Риск** – последствия катастрофические
  - Тестировщики и разработчики **совместно** снижают риски

SW компании

«просветленные» SW компании

ИТМО ВТ

Тестирование и качество ПО

- Уровень 4 — Тестирование - это возможный способ оценки качества программного обеспечения в терминах найденных дефектов
  - Функциональное
  - Нефункциональное :надежность, практичность, эффективность, сопровождаемость и переносимость
  -
- ISO 9126 «Информационная технология. Оценка программного продукта»

Традиционные производства

ИТМО ВТ

Тестирование и качество

- Другие способы оценки качества
  - Разработка стандартов
  - Обучение
  - Анализ дефектов

ИТМО ВТ

Цели тестирования (ISTQB)

- Цели тестирования:
  - Обнаружение дефектов
  - Повышение уверенности в уровне качества
  - Предоставление информации для принятия решений
  - Предотвращение дефектов

ИТМО ВТ

Цели тестирования

- Увеличение приемлемого уровня пользовательского доверия в том, что программа функционирует корректно во всех необходимых обстоятельствах
  - Корректное поведение
  - Уровень доверия
  - Необходимые обстоятельства - требование реального окружения

ИТМО ВТ

Корректное поведение

- Необходимо определение
  - Из требований
  - Спецификаций, описаний, …
  - Зависит от уровня тестирования

ИТМО ВТ

Уровень доверия

- Наглядность
- Уровень остаточного обнаружения дефектов
  - Число дефектов обнаруженных тестом или набором тестов
  - Число дефектов обнаруженных в заданное время

«Меньше 10-ти критических дефектов найдено за последние 7 дней»

- Требования к надежности
  - Сложно показать без испытаний, т. е. работающего ПО

Среднее время между отказами не должно быть меньше 5000 часов

ИТМО ВТ

Реальное окружение

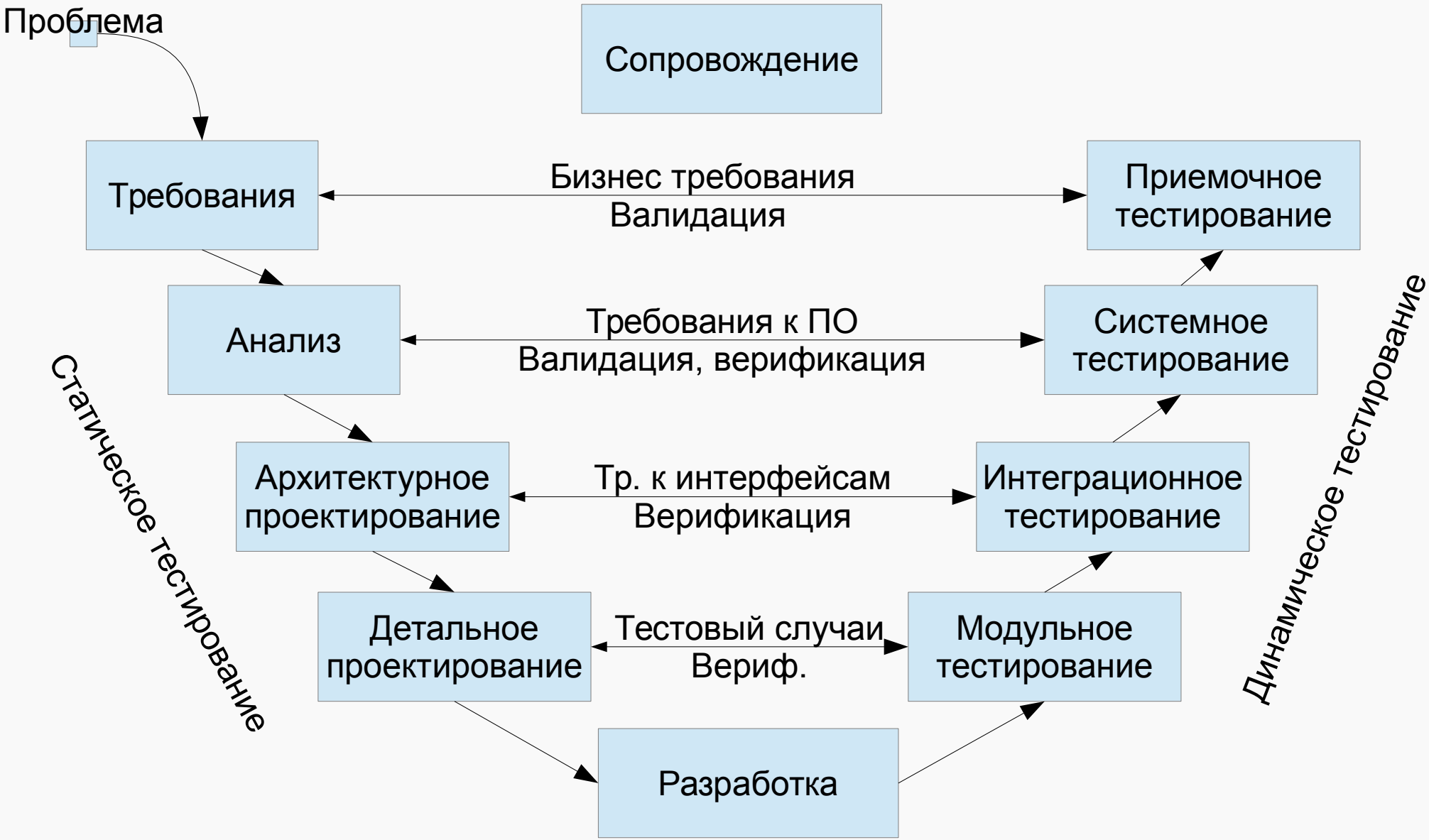
- Реалистичное количество данных - таких же как в целевой системе
  - В университете 5000 студентов, небольшой рост
  - Необходим тест на 5000, 6000, 7000 студентов, но не на 100000
- Реалистичный набор, комбинация входных данных

ИТМО ВТ

Полное тестовое покрытие

- public long multiply (int A, int B)
  - Как протестировать?
  - Сколько протребует памяти?
  - Сколько будет выполнятся на 3Ггц ЦПУ?

Традиционная V-model



ИТМО ВТ

Статическое и динамическое тестирование

- Статическое (рецензирование)
  - Не включает выполнения кода
  - Ручное, автоматизированное
  - Неформальное, сквозной контроль, инспекция
- Динамическое
  - Запуск модулей, групп модулей, всей системы
  - После появления первого кода (а иногда перед!)

ИТМО ВТ

Валидация и Верификация

- Валидация
  - Проверка на соответствие ожиданиями
  - ПО выполняет требования пользователя?
  - Пирожок (мясной, вегетарианский, сладкий)
  - Have we done the right thing?
- Верификация
  - Внутреннее управление качеством
  - ПО выполняет требования спецификации?
  - Пирожок (Размер, степень прожарки, начинка, …)
  - Have we done the thing right?

ИТМО ВТ

Источники данных для тестов

- Описания ПО — метод «черного ящика»
  - Спецификации, требования, дизайн.
  - Запуск и сравнение результатов с эталоном
- Исходный код — метод «белого ящика»
  - Переходы, утверждения, условия...
  - Анализ путей, структуры
- Опыт
- Модели
  - UML

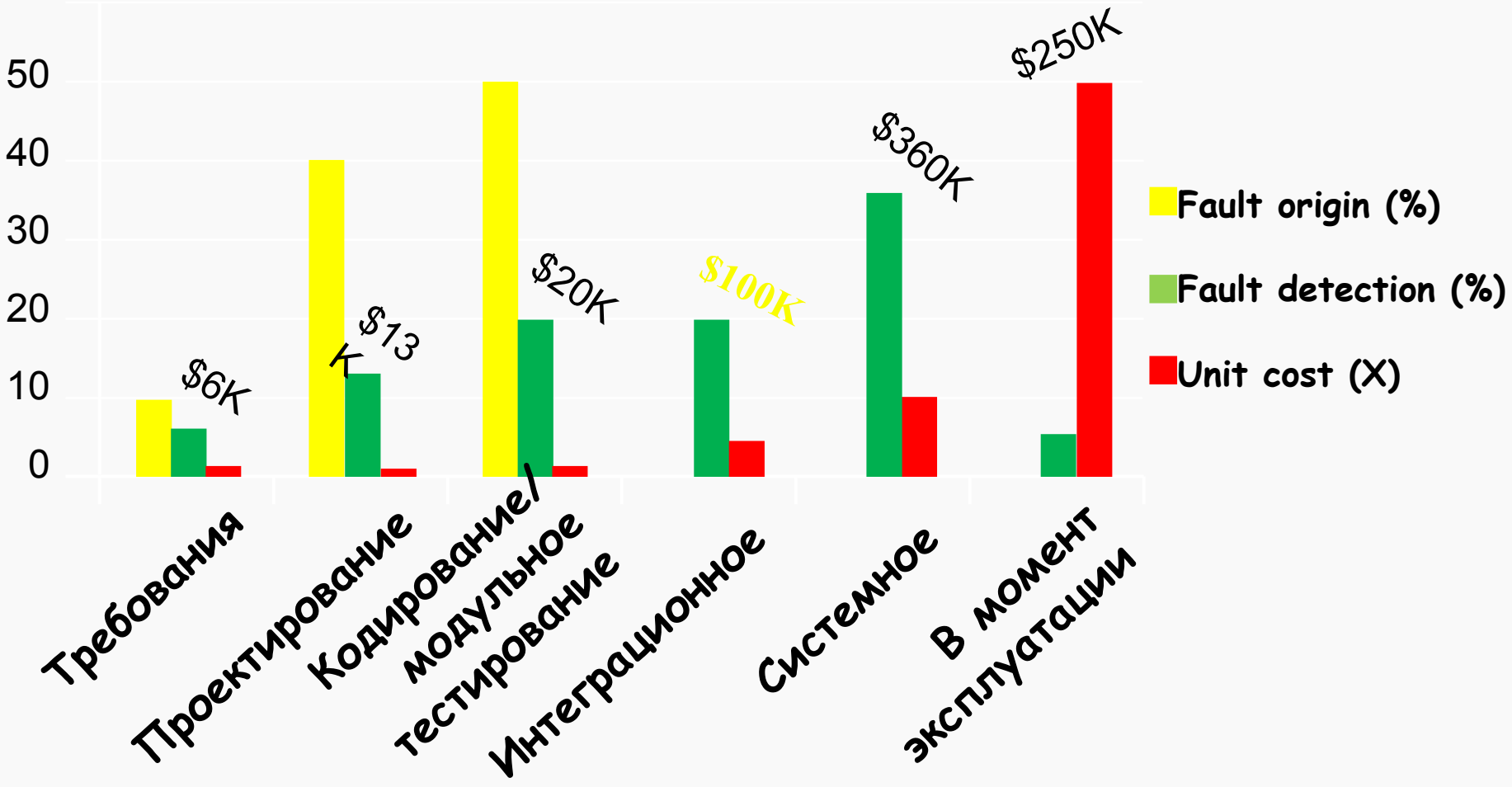
ИТМО ВТ

Деятельность и роли в тестировании

- Проектирование тестов
  - На основании формальных критериев
  - На основании знаний предметной области, опыта и экспертизы
- Автоматизация тестов
  - Знание средств, скриптов
- Исполнение тестов
  - Нет специальных требований к квалификации
- Анализ результатов
  - Знания предметной области

ИТМО ВТ

Цена позднего тестирования



Software Engineering Institute; Carnegie Mellon University; Handbook CMU/SEI-96-HB-002

ИТМО ВТ

Принципы тестирования (ISTQB)

1. Тестирование демонстрирует наличие дефектов
2. Исчерпывающее тестирование недостижимо
3. Раннее тестирование
4. Скопление дефектов
5. Парадокс пестицида
6. Тестирование зависит от контекста
7. Заблуждение об отсутствии ошибок.

ИТМО ВТ

Основы тестирования

2

ИТМО ВТ

Определения

- Отладка (debugging)
- Требование (requirement)
- Тестовый случай/сценарий (test case/scenario)
- Цель тестирования (test target)

ИТМО ВТ

Тестовый случай
Input → Processing → Output

- Входные значение
  - Данные или управляющие воздействия
- Предусловия, условия выполнения, постусловия
- Ожидаемый результат
  - Выходные данные и состояния, изменения в них, и другие последствия теста
  - Определен до запуска теста! (в идеале и TDD)

ИТМО ВТ

Тестовый случай (2)

- Повторяемый, автоматизируемый
- Учитывает состояния (если есть)
  - Переходы между состояниями
    - Правильные: корректный результат
    - Неправильные : корректные сообщения об ошибках
- В российской официальной терминологии используется термин сценарий

ИТМО ВТ

Тестовый сценарий

- Последовательность случаев
  - Типичное использование системы

№	Начальное состояние	Ввод	Действие системы	Вывод	Конечное состояние
1	Готов	Пользователь вставляет карточку	Успешное чтение карточки	Приглашение “введите pin”	Ожидание pin-кода
2	Ожидание pin-кода	Вводим **верный** pin-код	Проверка pin-кода	Приглашение к выбору транзакции	Ожидание выбора транзацкии
3	Ожидание выбора транзацкии	Выбор выдачи 5000 рублей	Проверка баланса, возможности выдачи	Деньги	Выдача денег
4	Выдача денег	Пользователь берет деньги и карточку	Завершение выдачи	Благодарность за использование	Готов

ИТМО ВТ

Тестовые сценарии (2)

- Должны обрабатывать
  - Корректное поведение и вариант ошибки

№	Начальное состояние	Ввод	Действие системы	Вывод	Конечное состояние
1	Готов	Пользователь вставляет карточку	Успешное чтение карточки	Приглашение “введите pin”	Ожидание pin-кода
2	Ожидание pin-кода	Вводим **неверный** pin-код	Проверка pin-кода	Сообщение об ошибке	Ожидание pin-кода
3	Ожидание pin-кода	Вводим **неверный** pin-код	Проверка pin-кода	Сообщение об ошибке	Ожидание pin-кода
4	Ожидание pin-кода	Вводим **неверный!!!** pin-код	Проверка pin-кода, блокировка карточки	Сообщение об ошибке и блокировка	Готов

ИТМО ВТ

Тестовые сценарии (3)

- Определение корректного поведения в:
  - Требованиях (системное, приемочное тестирование)
  - Архитектуре (интеграционное тестирование)
  - Проектных документах (модульное тестирование)
- Тестовые сценарии
  - Можно взять из документации к проекту: Use-case → Test-case

ИТМО ВТ

Сколько тестов?

- Требуется баланс
  - Много тестов → больше покрытие → качество выше
  - Меньше тестов → выше скорость разработки → быстрее выход на рынок
- Необходимо выбрать специфические значения для тестирования
  - Нельзя же тестировать вечность!
  - Полное покрытие недостижимо

ИТМО ВТ

Выбор тестового покрытия

- Эквивалентное разбиение (партиции эквивалентности)
  - Анализ граничных значений
- Таблица решений (альтернатив)
- Таблицы переходов
- Сценарии использования

ИТМО ВТ

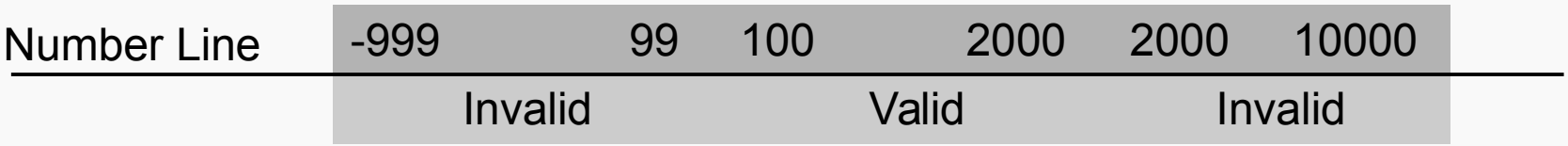
Анализ эквивалентности

- Набор входных данных на которых результат является эквивалентным или подобным
  - Например, если вывод системы одинаковый — находимся внутри эквивалентной области
  - Дополнительно может использоваться анализ граничных значений
- Банкомат:
  - Купюры по $100
  - За раз максимум $2000
  - За день $10000
  - Комиссия 1% но не меньше $5

Анализ эквивалентности (2)

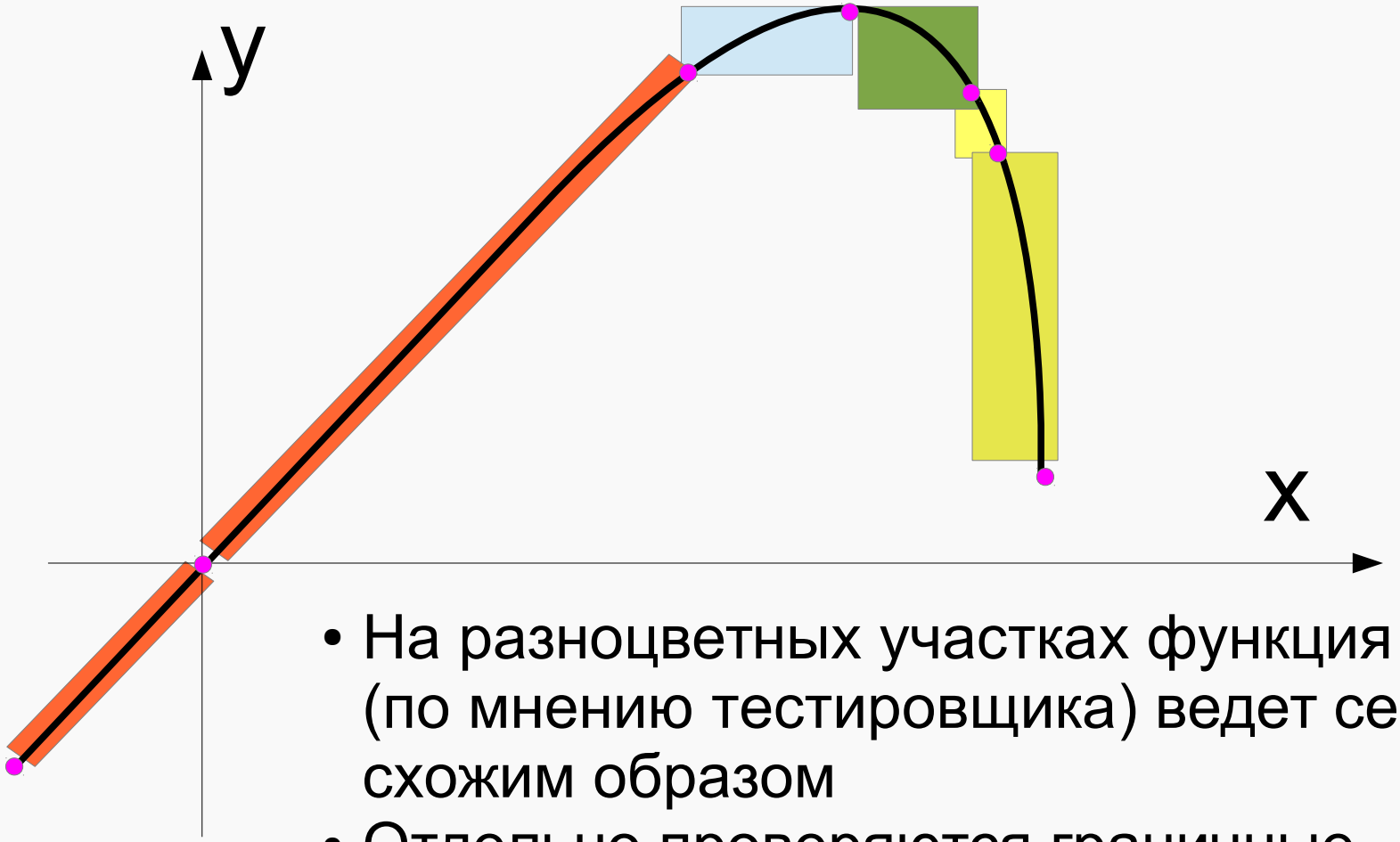
Сумма к снятию	Сумма к списанию	Результат	Класс результата
-1	-	нет	невалидный
1..99,101..199, ...	-	нет	невалидный
100,200,..,500	105,205, …, 505	выдача	валидный
600,700,...,2000	606,707, …, 2020	выдача	валидный
2100,2200...	-	нет	невалидный
2000+1..99	-	нет	невалидный
2000+100	2020+105	Выдача в два этапа	валидный
2000+2000+2000+ 2000+2000	10100	Выдача в 5 этапов	валидный

Граничные значения для параметра выдачи средств



Анализ эквивалентности

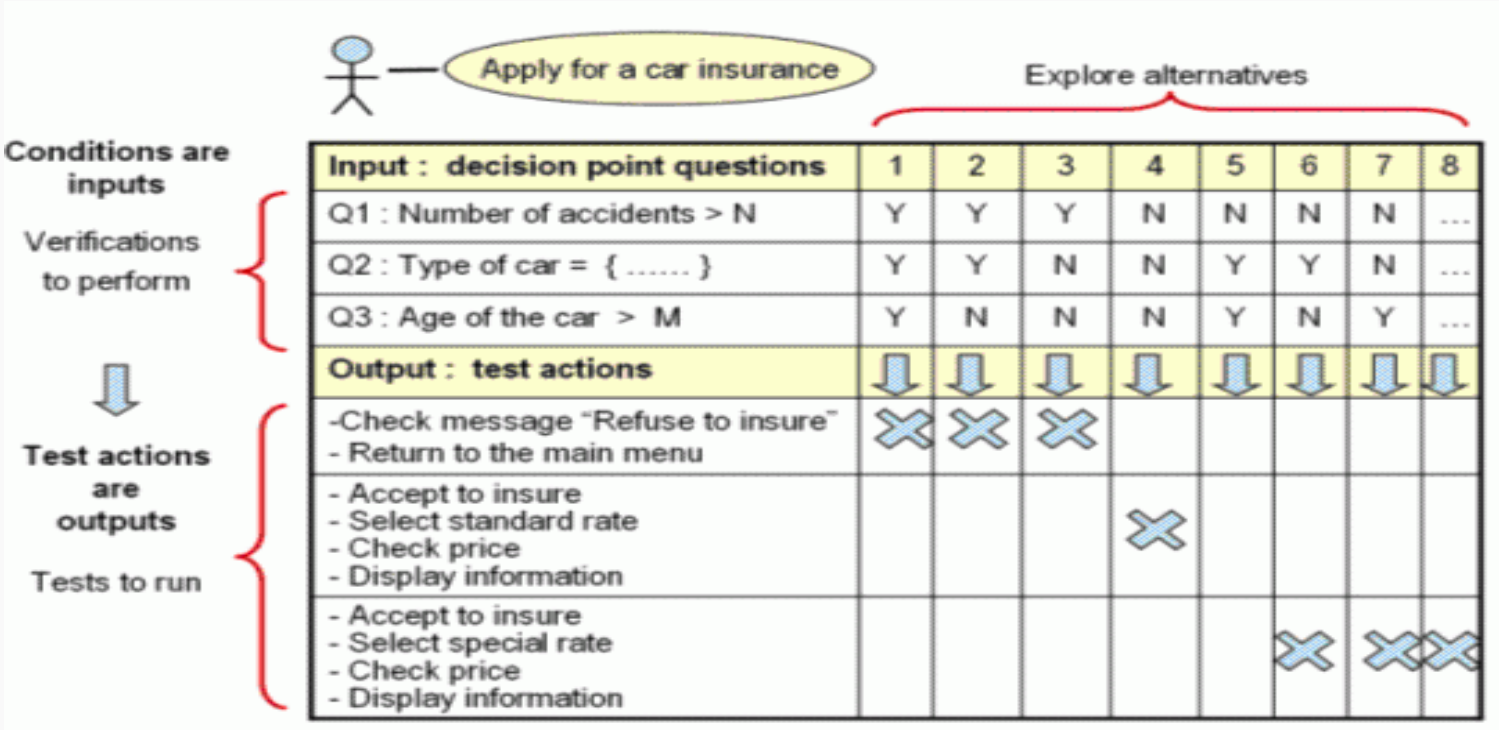
Зависимость удовлетворения от количества съеденного холодца



- На разноцветных участках функция (по мнению тестировщика) ведет себя схожим образом
- Отдельно проверяются граничные значения

Таблица решений

- Используется в системах со сложной логикой, описание конечного автомата
- Может включать большой набор условий (обычно true-false), и действий

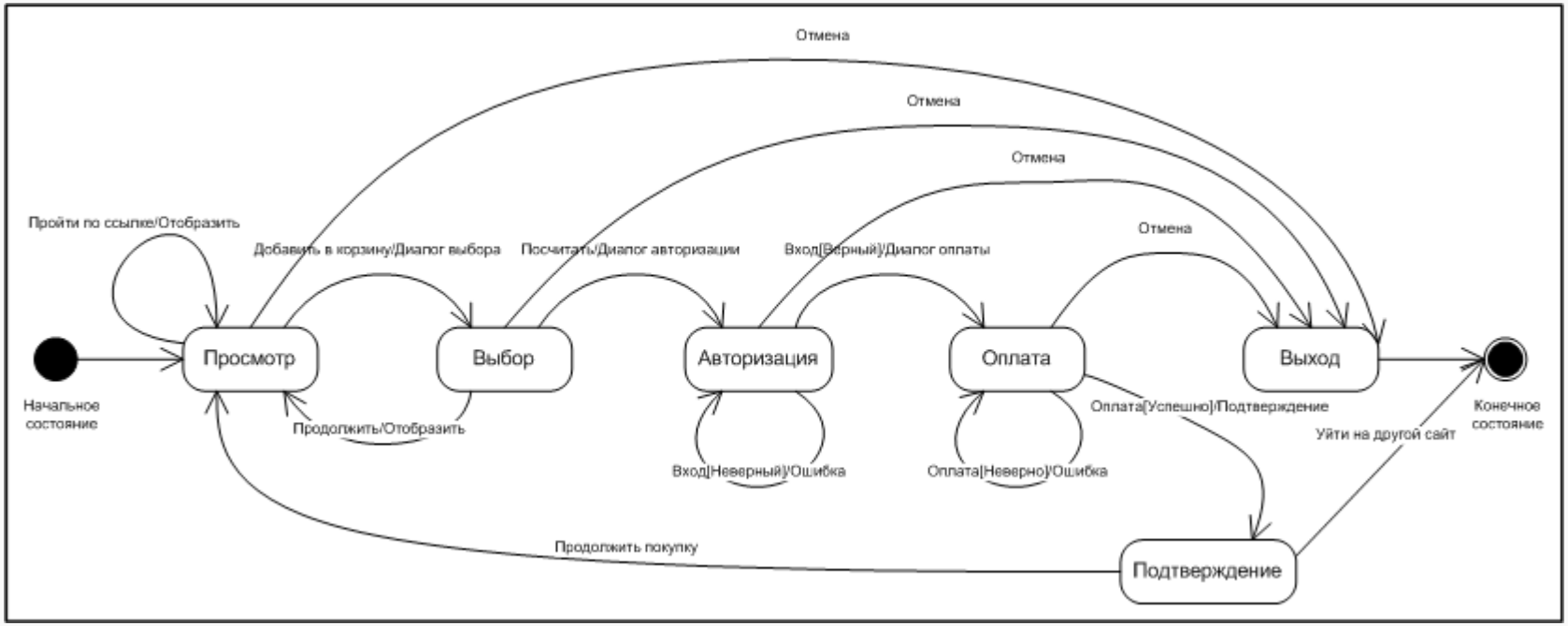


Input : decision point questions	1	2	3	4	5	6	7	8
Q1 : Number of accidents > N	Y	Y	Y	N	N	N	N	...
Q2 : Type of car = { ..... }	Y	Y	N	N	Y	Y	N	...
Q3 : Age of the car > M	Y	N	N	N	Y	N	Y	...
Output : test actions	⇩	⇩	⇩	⇩	⇩	⇩	⇩	⇩
-Check message "Refuse to insure" - Return to the main menu	X	X	X					
- Accept to insure - Select standard rate - Check price - Display information				X				
- Accept to insure - Select special rate - Check price - Display information						X	X	X

http://software-testing.ru/library/testing/functional-testing/129--functional-tester-ibm-rational-

Таблицы переходов

- Позволяют выбрать состояния и их комбинации, которые можно опустить
  - Рекс Блэк «Advanced Software Testing».
  - http://inrecolan.ru/blog/viewpost/361



http://inrecolan.ru/blog/viewpost/360

http://inrecolan.ru/blog/viewpost/361

Таблицы переходов (2)

- Прокрыть определенные строки тестами
- while (есть тесты с «непокрытыми» строками)
  - Выбрать состояния «не определено»
  - Попытаться покрыть тестом

Состояния		События/Условия		Текущее состояние	Событие/Условие	Действие	Новое состояние
				Просмотр	Пройти по ссылке	Отобразить	Просмотр
		Пройти по ссылке		Просмотр	Добавить в корзину	Диалог выбора	Выбор
		Добавить в корзину		**Просмотр**	**Продолжить**	**Не определено**	**Не определено**
		Продолжить		**Просмотр**	**Выписать**	**Не определено**	**Не определено**
Просмотр		Выписать		**Просмотр**	**Вход [неверный]**	**Не определено**	**Не определено**
Выбор		Вход [неверный]		**Просмотр**	**Вход [верный]**	**Не определено**	**Не определено**
Авторизация	X	Вход [верный]	=	**Просмотр**	**Оплата [неверно]**	**Не определено**	**Не определено**
Оплата		Оплата [неверно]		**Просмотр**	**Оплата [успешно]**	**Не определено**	**Не определено**
Подтверждение		Оплата [успешно]		**Просмотр**	**Отмена**	**Нет действия**	**Выйти**
Выйти		Отмена		**Просмотр**	**Продолжить покупку**	**Не определено**	**Не определено**
		Продолжить покупку		**Просмотр**	**Уйти на другой сайт**	**Не определено**	**Не определено**
		Уйти на другой сайт		**Выбор**	**Пройти по ссылке**	**Не определено**	**Не определено**
				{53 строки, сгенерированных по шаблону, не показаны}			
				Выйти	**Уйти на другой сайт**	**Не определено**	**Не определено**

http://inrecolan.ru/blog/viewpost/361

Сценарии использования (функциональное тестирование)

Прецедент: ViewInformationAboutTheRoute
ID: 2
Краткое описание: Пользователь просматривает информацию о маршруте
Главные актёры: Клиент (или любой другой пользователь)
Второстепенные актёры: нет
Предусловия: Пользователю выведен список маршрутов. Пользователь может быть не авторизован в системе
Основной поток: Прецедент начинается когда Пользователь выбирает конкретный маршрут Система отображает подробную информацию по выбранному маршруту

- Выбираем сценарий
- Проверяем его

ИТМО ВТ

Автоматизация тестов

- Регрессионное тестирование
- Повторение тестового сценария
- Приемочное тестирование
- Сокращение ручного труда?
- Проверка одного приложения в разных окружениях

ИТМО ВТ

Модульное и интеграционное тестирование

3

ИТМО ВТ

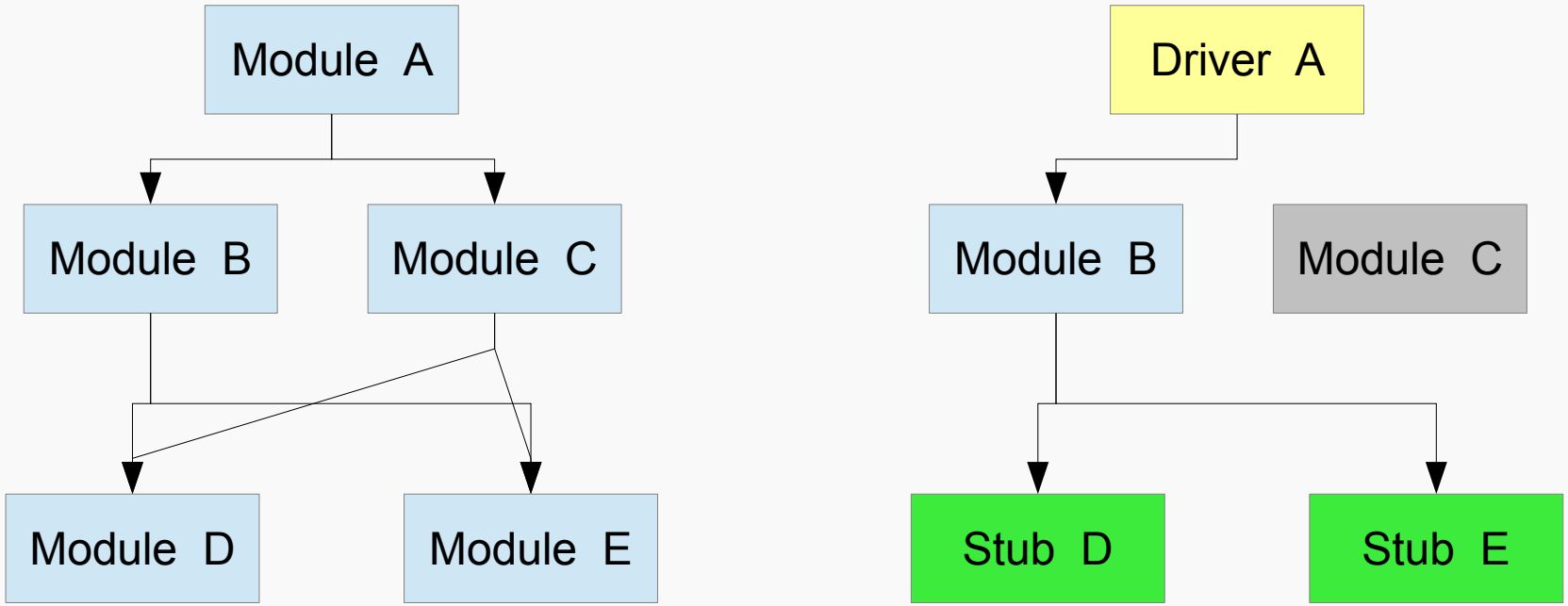
Модуль

- Модульное (компонентное) тестирование - тестирование отдельных компонентов программного обеспечения [IEEE 610].
  - Метод или класс
  - Программный модуль
- Модули описаны в дизайне
- Необходимо изолировать модуль из системы

ИТМО ВТ

Изолирование модулей

- Драйвер вместо вызываемой
- Заглушка вместо подчиненной



ИТМО ВТ

Заглушка

- Эмулирует поведение подчиненной программы
  - Подпрограмма, функция, процедура
  - Аппаратное прерывание, передача данных
- Интерфейс совпадает, внутренность нет
  - Предопределенные ответы для заданных аргументов, исключения
  - Постоянная генерация прерываний
- Используются вместо реальной программы
  - Компилируется и линкуется

ИТМО ВТ

Заглушка (2)

- Простая
  - Нельзя тестировать заглушку!
- Обычно 1 строка кода
- Возможна дополнительная логика
  - 50 раз возвращать 42, на 51 — IndexOutOfBounds
- Может читать значения из текстового файла
- Возможна настройка драйвером перед выполнением

ИТМО ВТ

Драйверы

- Эмулирует вызываемый компонент
- Обычно уже более сложная программа
  - Устанавливает окружение
  - Подготавливает входные данные
- Дополнительно может:
  - Запускать серию тестов
  - Настраивать заглушки
  - Формировать журнал результатов

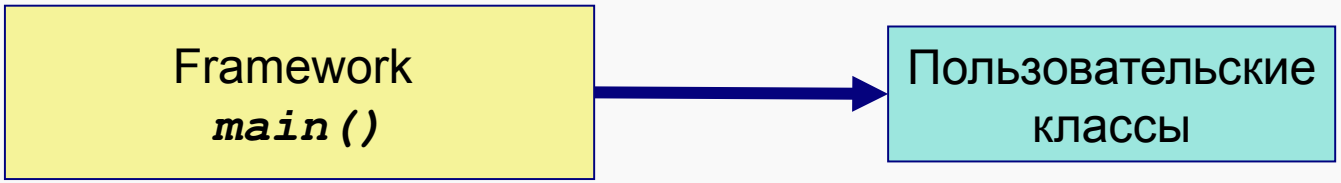
ИТМО ВТ

Test Fail

- Для анализа результата тестов программисту необходимо:
  - Входные и выходные значения
  - Значения измененных переменных
  - Пройденные пути, принятые решения
  - Симптомы сбоя
- Где ошибка в тесте или в ПО?

Фреймворки для модульного тестирования

- XXXunit — много разных!
- Фреймворк vs Библиотека



ИТМО ВТ

Junit 4

- JUnit фреймворк обеспечивает:
  - Аннотации для маркировки метода как теста @Test
  - Аннотации для маркировки действий до и после теста @Before, @After, @BeforeClass, @AfterClass
  - Методы для проверки (assertion)
  - UI, журнал тестов...

ИТМО ВТ

Пример

```
import org.junit.*;
import static org.junit.Assert.*;
public class MoneyTest {
   private Whale whale;
   @Before public void setUp() {
         whale = new Whale();
           whale.setLocation(”Где-то высоко”);
   }
   @Test public void testDown() {
        whale.fallDown();
        assertEquals(“Кит не упал”,
                          “на земле”,
                         whale.getLocation());
   }
}
```

ИТМО ВТ

Дополнительные возможности

```
@Test (timeout=10)
 public void longLoop() {
    assertEquals(PI.computeNNumber(1E10) ,3);
}
@Test (expected=IllegalArgumentException.class)
public void testSqrt() {
    Math.Sqrt(-5);
}
```

@RunWith(value=Parameterized.class) и @Parameters

ИТМО ВТ

@Ignore и автобоксинг

```
long sum(long x, long y) { return x + y; }


@Ignore("Евгений, помогите!")
@Test
public void add() {
    assertEquals(4, program.sum(2, 2));
}
```

expected: <4> but was: <4>

ИТМО ВТ

Автобоксинг

- В JUnit4 assertEquals не существует для примитивных типов
  - В Тесте 4 is упакуется в Integer, sum возвращает long
- Сообщение об ошибке значит:
  - expected int 4, but got long 4
- Надо исправить 4 to a 4L
  - assertEquals(4L, program.sum(2, 2));

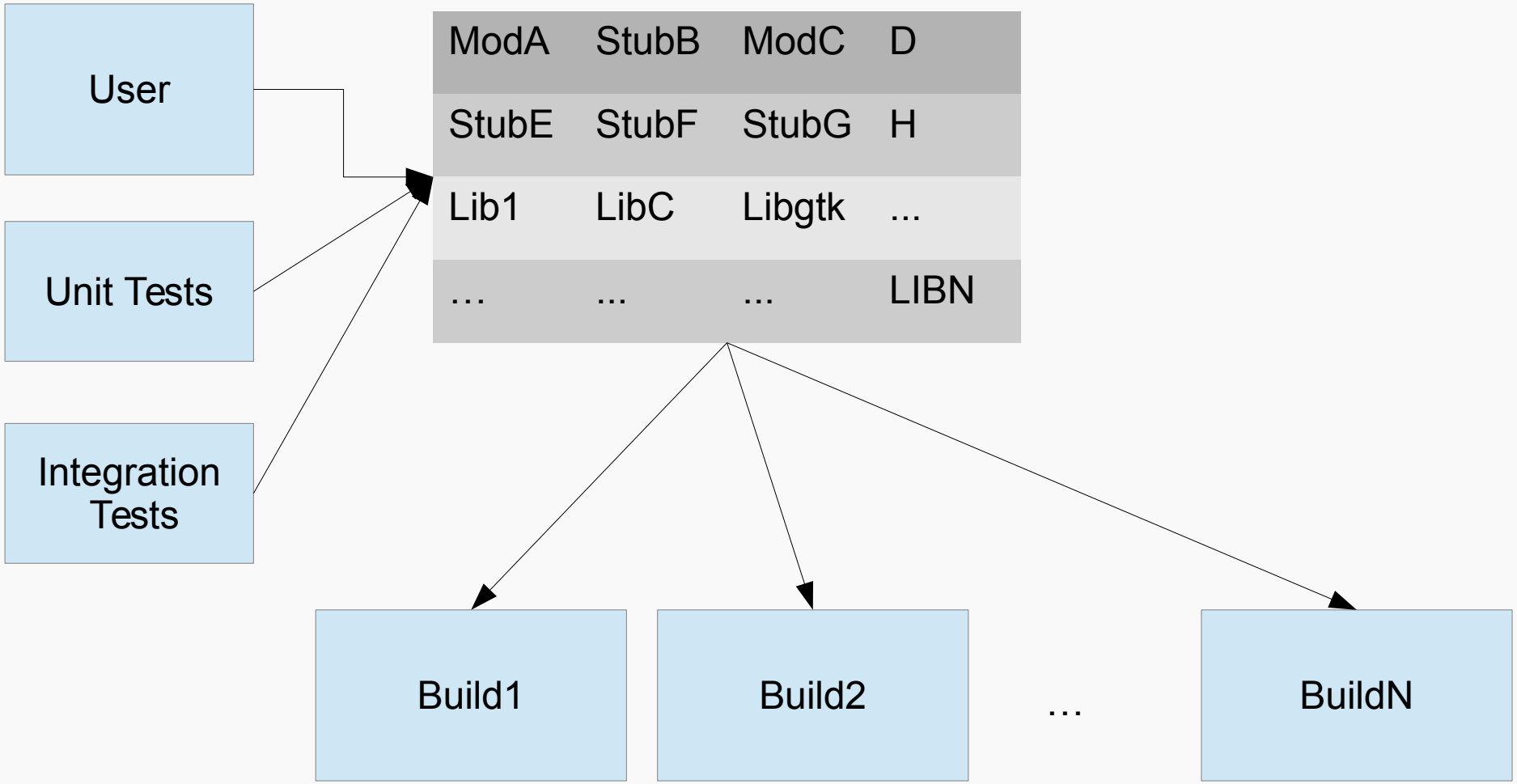
ИТМО ВТ

Интеграционное тестирование

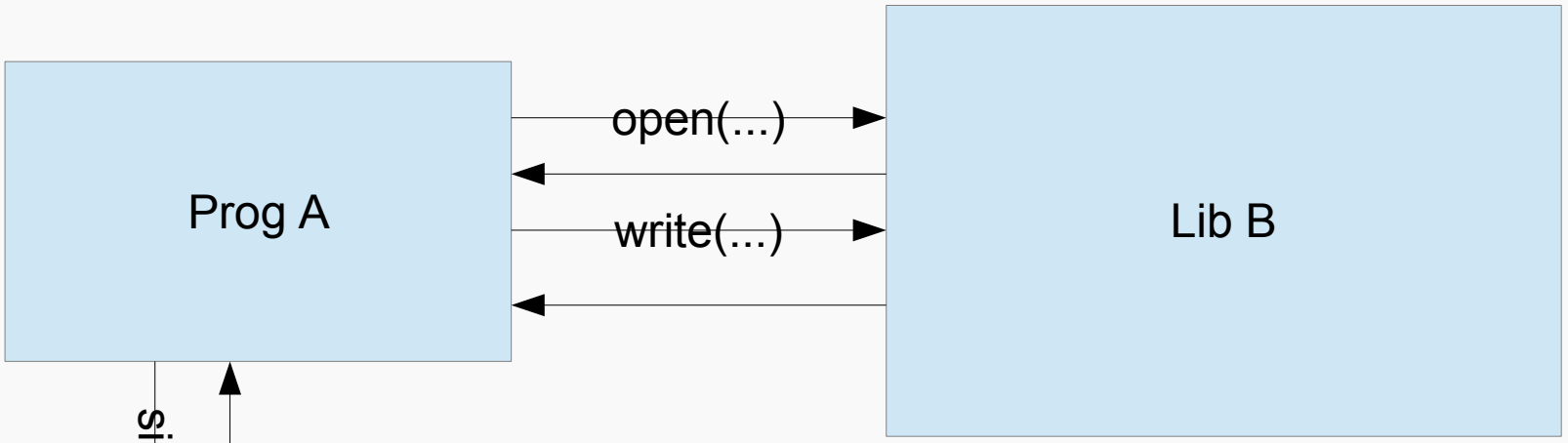
- Проверяет интерфейсы и взаимодействие модулей (компонент) или систем
  - Вызовы API, сообщения между ОО компонентами
  - Баз Данных, пользовательский графический интерфейс
  - Интерфейсы взаимодействия (сетевые, аппаратные, локальные, ….)
  - Инфраструктурные
- Может проводиться когда два компонента разработаны (спроектированы)
  - Остальные добавляются по готовности

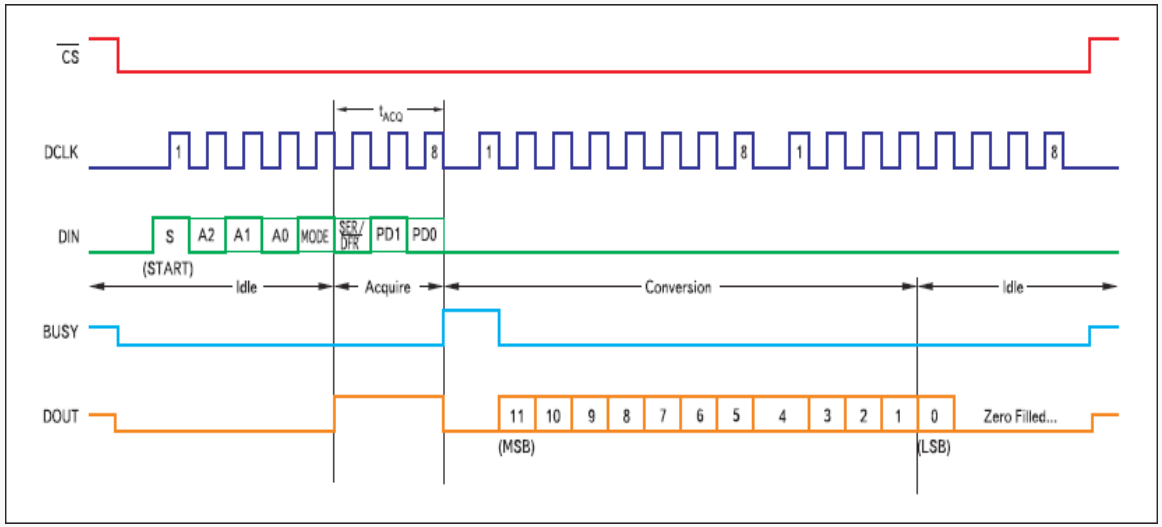
ИТМО ВТ

Интеграция

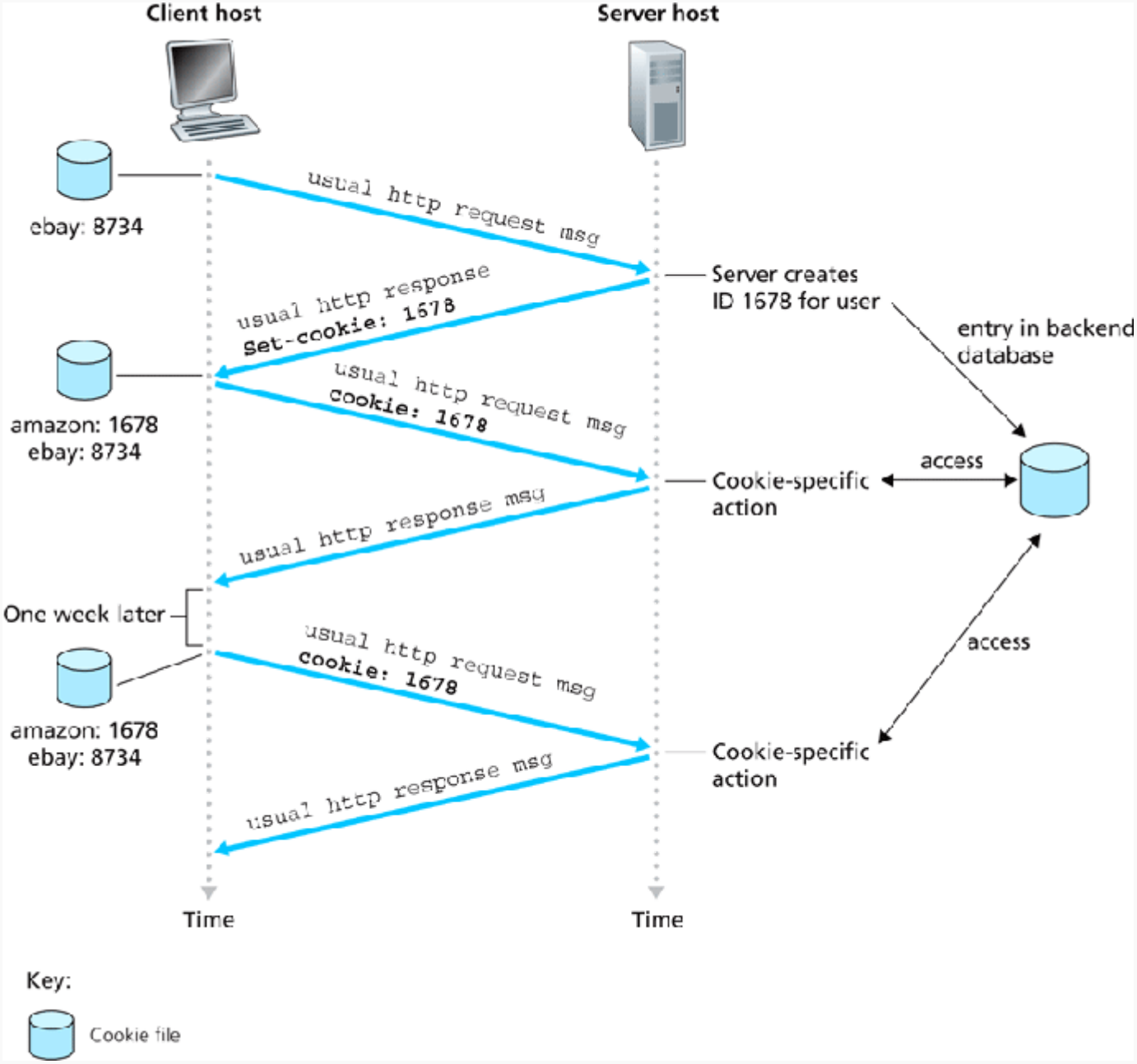


Вызовы и сообщения



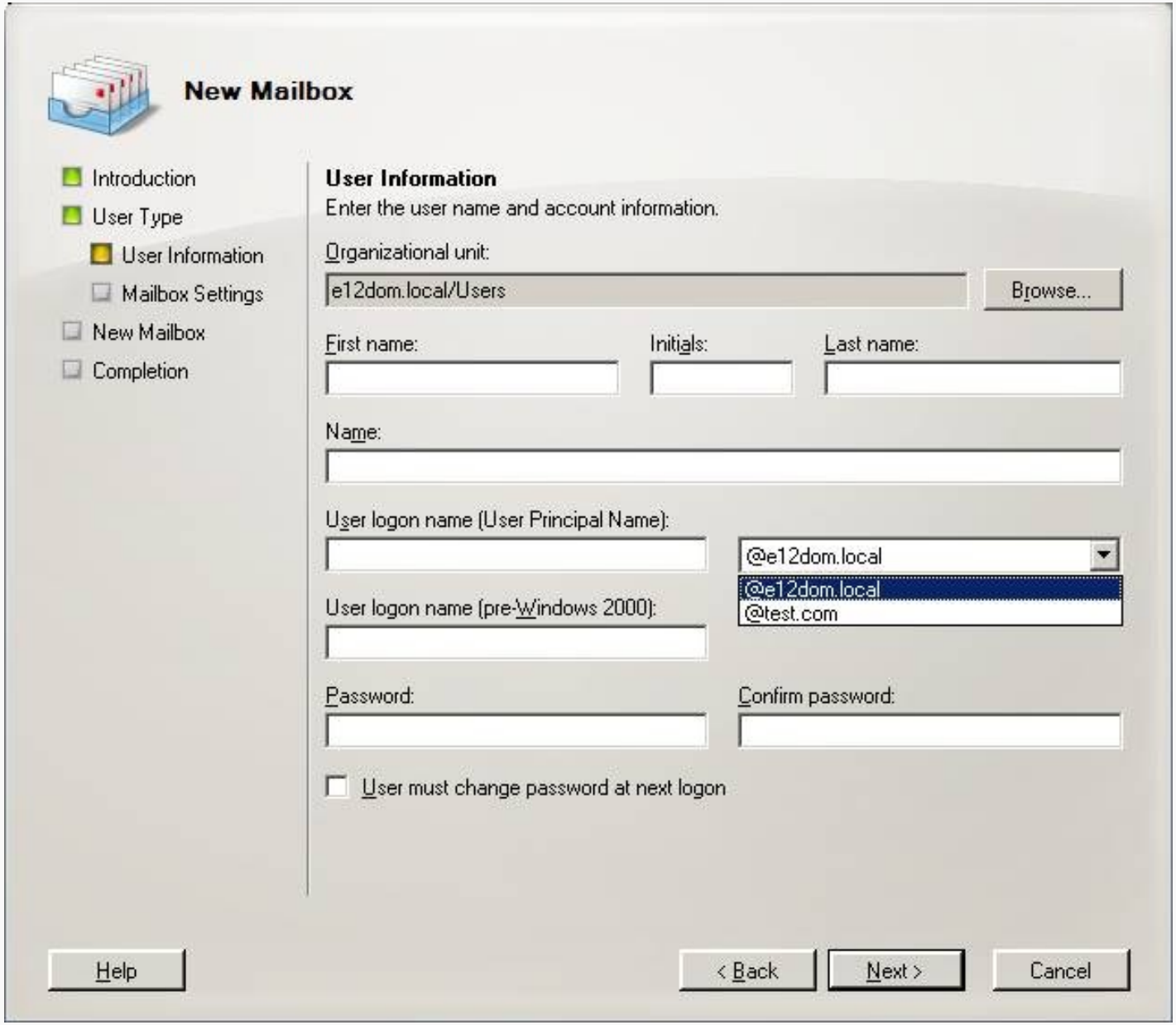


Сетевое взаимодействие



ИТМО ВТ

Пользовательские интерфейсы

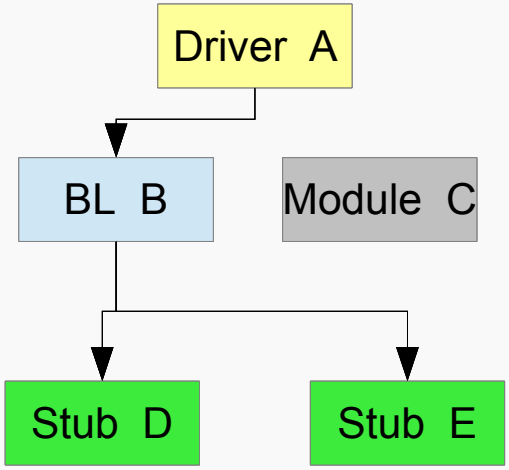


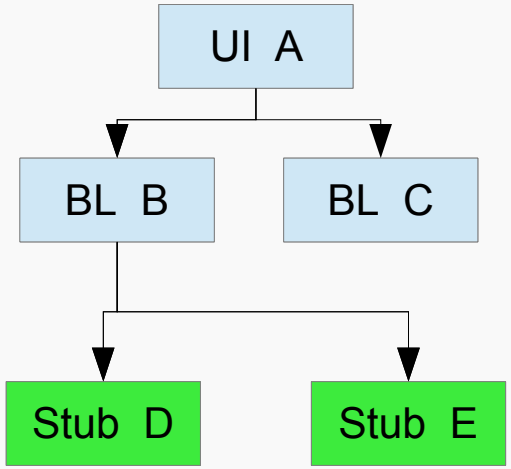
ИТМО ВТ

Стратегии интеграции

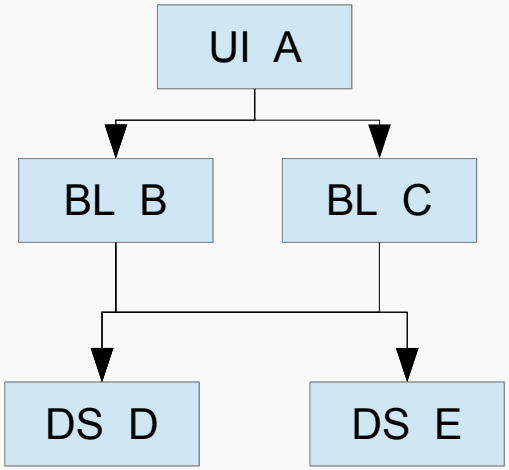
- Больше объем интеграции — больше сложность
  - Для каждого интерфейса должен быть разработан короткий тест план
- Выбор в зависимости от архитектуры ПО
- Последовательность имеет значение
- Можно тестировать нефункциональные характеристики

Сверху вниз





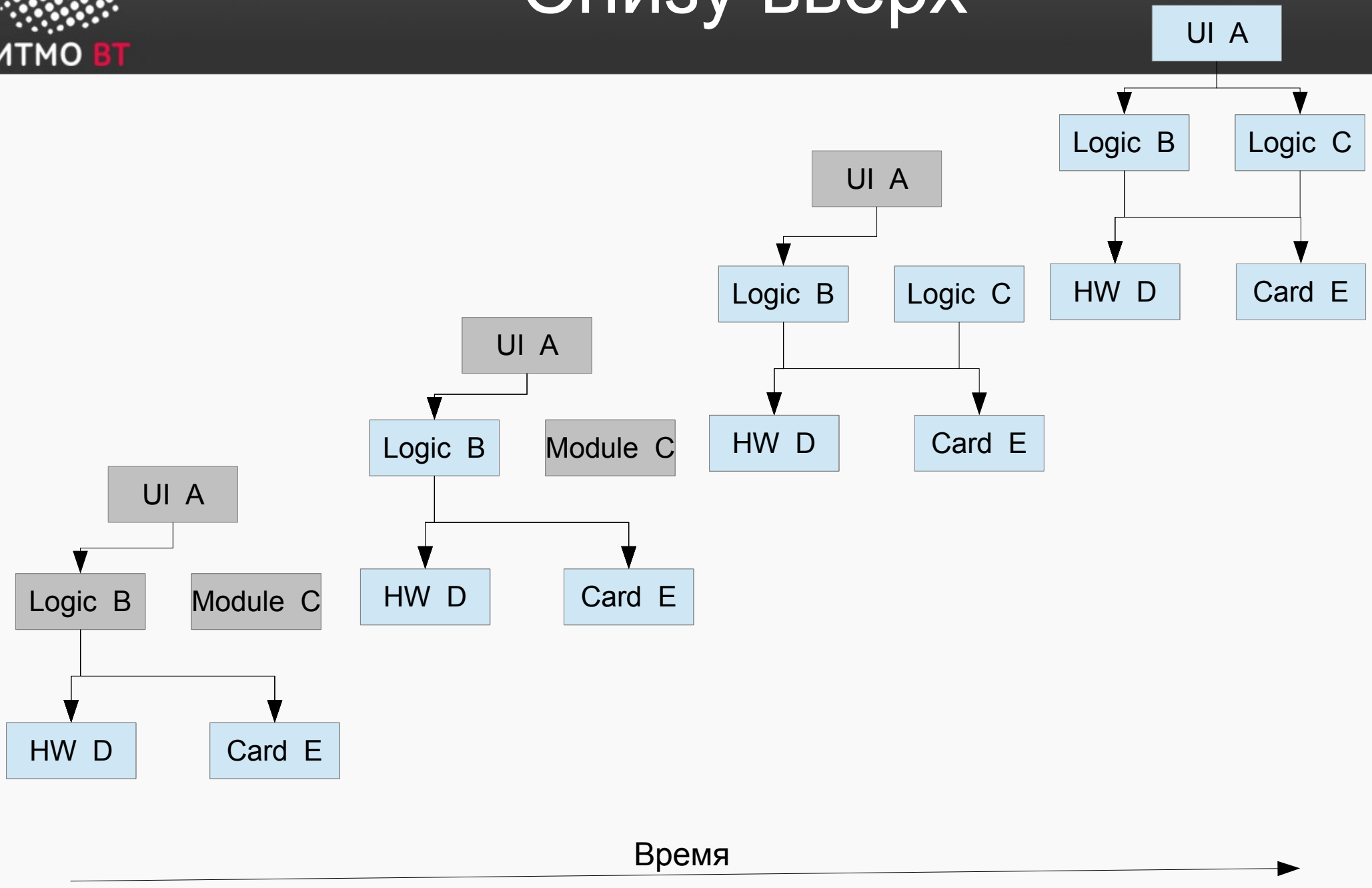
BL — бизнес логика
UI — интерфейс пользователя
DS — уровень хранения



\+ Быстро появляется «осязаемое» приложение
\- Большое количество заглушек

ИТМО ВТ

Снизу вверх



ИТМО ВТ

Другие

- Функциональная (end to end) — по одной функции
  - Собрали 1 сценарий UI-Логика-БД, добавили еще один такой-же
- Ядро (backbone)
  - Экран, клавиатура, мышь работают с минимальными функционалом
  - Добавить цвета на экран, колесо прокрутки …
- Большой взрыв (big bang)
  - Собрать все вместе и молиться

ИТМО ВТ

Основные принципы

- Интегрировать в первую очередь наиболее сложные компоненты
  - Снизит риски
  - Больше времени для исправления ошибок
- Выбирать порядок интеграции с учетом порядка разработки
- Использовать регрессионное тестирование после каждого этапа интеграции
- Автоматизировать тесты

ИТМО ВТ

Функциональное тестирование

- На базе сценариев использования
- Ручное/автоматическое
- На готовой системе, в рамках модульного и интеграционного
- Проверяются функции системы начиная с интерфейса пользователя
- Средства автоматизации
  - Открытые: Selenium, Sahi, Watir
  - Коммерческие: от HP, Rational (IBM) ….

ИТМО ВТ

Selenium

- Набор средств автоматизации
  - IDE
  - Selenium Server / WebDriver
  - Grid
- Тестирование web-приложений
- Кросс- браузерный
- Разработка сценариев на многих языках
  - java, php, python, c#
- Встроенные конструкции assert
- Встроенный механизм логирования ошибок

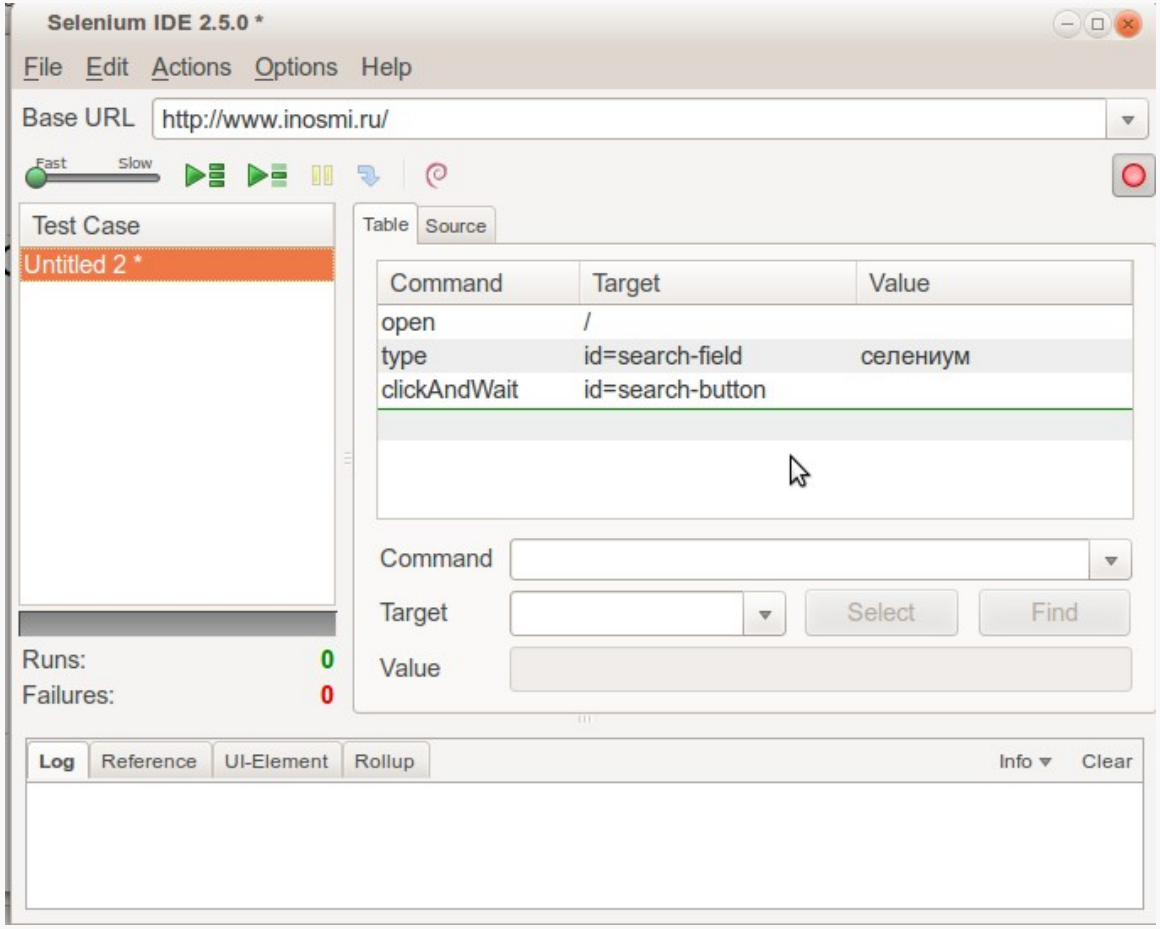
ИТМО ВТ

Selenium IDE

- Интегрированная среда исполнения и разработки тестов
- Разработано как расширение Firefox
- Запись тестов в виде Java, Ruby, HTML
- Добавление assert и команд

ИТМО ВТ

Selenium IDE



ИТМО ВТ

Команды

- click или clickAndWait — ссылки, переключатели, radio-кнопки
- type — ввод значений
- select — выбор значений из списка
- open — открывает страницу
- assert***
- wait*** - ожидание события
- verify *** - проверка

ИТМО ВТ

Assertion & Verification

- Предназначены для проверки содержимого элемента UI
  - Элемент присутствует?
  - Есть ли необходимый текст на странице?
- Если verification неуспешна — тест продолжается
- Если неуспешна assertion — тест останавливается

ИТМО ВТ

Добавление

- Добавляется во время записи теста щелчком правой кнопки мыши на элементе
  - verifyTextPresent
  - verifyTitle
  - verifyElementPresent
  - verifyValue
- Assertion - аналогично

ИТМО ВТ

Синхронизация или WaitFor Command

- waitForPageLoad(timeout) загрузка страницы ошибка по таймауту
- waitForAlert
- waitForTable — полная загрузка таблицы
- waitForTitle — загрузка заголовка
- Другие команды синхронизации

ИТМО ВТ

Другие команды

- store — сохранение значений в переменной
- echo — запись значения в лог selenium
  - Можно использовать ${var}

Запись скрипта в виде кода (фрагмент)

```
public class HelloTest {
  private WebDriver driver;
  private String baseUrl;
  private boolean acceptNextAlert = true;
  private StringBuffer verificationErrors = new StringBuffer();

  @Before
  public void setUp() throws Exception {
    driver = new FirefoxDriver();
    baseUrl = "http://www.inosmi.ru/";
    driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);
  }

  @Test
  public void testHello() throws Exception {
    driver.get(baseUrl + "/");
    driver.findElement(By.id("search-field")).clear();
    driver.findElement(By.id("search-field")).sendKeys("селениум");
    driver.findElement(By.id("search-button")).click();
  }
```

•

ИТМО ВТ

Ограничения SeleniumIDE

- Запускает тесты только в Firefox
- Слабо развитое управление логикой теста (циклы, условия, ….)
- Запускает только свои сценарии
- Сложно использовать с динамическим содержимым

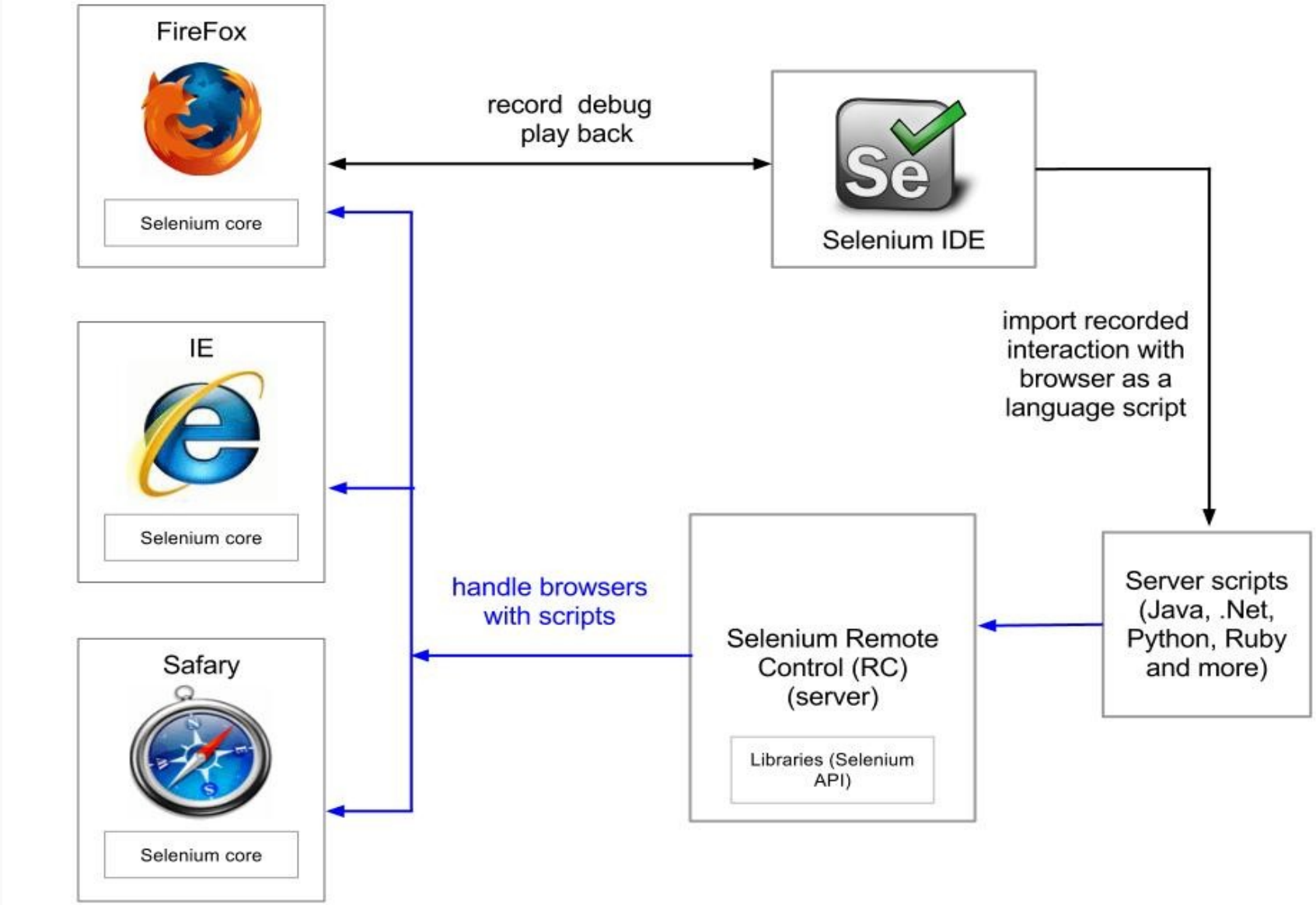
ИТМО ВТ

Selenium Server

- Может быть использован для кросс-браузерного тестирования
- Сервер для тестирования разработан на java
- Работает как прокси для web-запросов совместно с Webdriver для каждого браузера
- Используется совместно с многими языками программирования
- Разработка — плагины к NetBeans и Eclipse
- Запуск тестов — maven и ant

ИТМО ВТ

Принцип работы



ИТМО ВТ

Процесс разработки

- Записать сценарий в виде java кода
- Создать проект в IDE и добавить необходимые jar файлы (например)
  - junit-4.8.1.jar
  - selenium-java-client-driver.jar
  - selenium-server.jar
  - testng-5.12.jars
- Скопировать сценарий в IDE
- Запустить

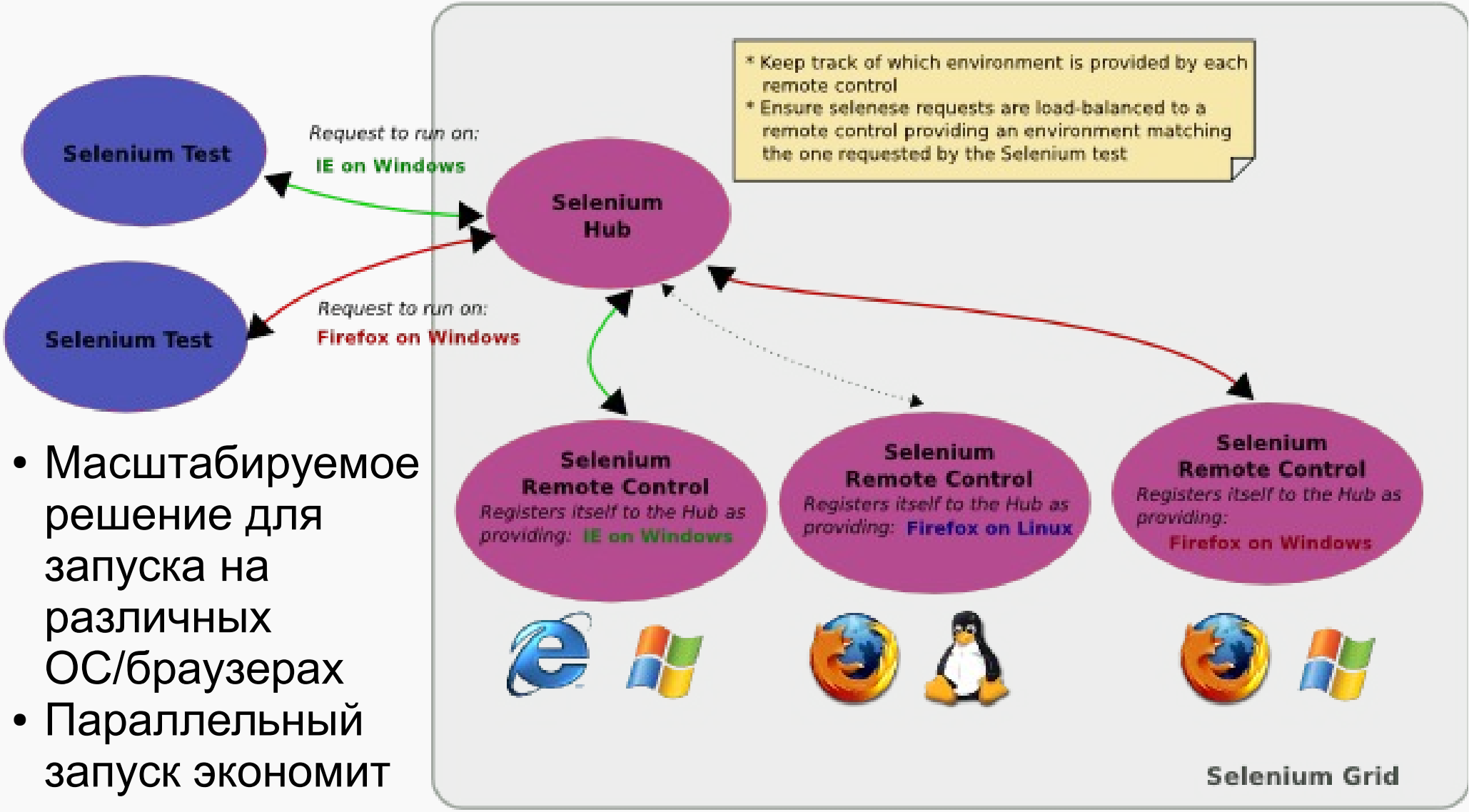
ИТМО ВТ

Xpath локатор

- В современных серверах приложений, порталах нельзя привязываться к компонентам по ID — он динамический
  - Используются классы стилей через xpath
- http://ru.wikipedia.org/wiki/XPath
- //div[contains(@class, 'article-heading')]

Selenium Grid

Selenium Grid : Requesting a Specific Environment



- Масштабируемое решение для запуска на различных ОС/браузерах
- Параллельный запуск экономит время

ИТМО ВТ

Автоматизация - не замена ручному тестированию

!!!

Статическое тестирование

4

ИТМО ВТ

Статическое тестирование

- Динамические тесты не могут быть исполнены, пока мы не создадим код
- Статические техники (IEEE 1028) могут быть применены до написания кода:
  - «Звонок другу» - неформальные ревью (рецензия)
  - Технические анализ (повторные просмотры)
  - Management review
  - Сквозной контроль
  - Инспекции

ИТМО ВТ

Статическое тестирование

- Может находить ошибки на ранних стадиях разработки!
- Снижение стоимости и рисков
  - Цена ошибки растет с фактором 10 в зависимости от стадии разработки продукта
- Объекты тестирования
  - Политики, стратегии, планы
  - Технические задания, спецификации
  - Артефакты со стадии проектирования, код
  - Планы и подходы к тестированию
  - Прочее...

Роли в формальном статическом тестировании

- Менеджер (ЛПР)
- Модератор
- Докладчик
- Автор
- Эксперты
- Секретарь

ИТМО ВТ

Неформальные рецензии

- Зачем? - Люди делают ошибки
- Взгляд со стороны — вторые мозги могут помочь найти ошибки
- Цель — найти технические проблемы
  - Не только опечатки!
  - Распространить артефакты, попросить коллег прочитать и комментировать
  - Обычно не очень эффективна — Почему?
  - Формальные методики более эффективны

ИТМО ВТ

Технический анализ

- Проверить продукт на соответствие и практическую пригодность
- Участники анализируют документы предварительно, подготавливают комментарии
- Анализ
  - Предлагает альтернативы и рекомендации
  - Формальный или неформальный

ИТМО ВТ

Сквозной контроль (Walkthrough)

- Проводится автором, который «ведет» аудиторию «через» артефакт
  - Или его части
- Может находить
  - Ошибки, аномалии, неэффективности
  - Спецификации, которые не могут быть проверены
  - Проблемы интерфейсов
  - Отклонения от практик кодирования
  - И пр.
- Может проводиться с образовательными целями

ИТМО ВТ

Инспекции

- 1972, IBM, Michael Fagan — формальный процесс
- Равноправный анализ (systematic peer evaluation)
- Цель — обнаружить и идентифицировать аномалии ПО
- Специально тренированный ведущий (не автор!):
  - Выбирает инспекторов
  - Распространяет заранее подготовленные документы
  - Ведет встречу
  - Убеждается в том, что принятые решения выполнены
  - Определяет и контролирует различные метрики

ИТМО ВТ

Инспекции - 2

- Четко определенные шаги
  - Вход
  - Планирование
  - Обзор
  - Подготовка
  - Обсуждение
  - Переработка
  - Выработка рекомендаций (follow up)
  - Выход

Могут повторяться

- Повторяется до тех пор, пока все участники включая модератора не удовлетворены

ИТМО ВТ

Инспекции — входные и выходные критерии

- Должны быть обязательно определены
  - Не тратить время попусту неподготовленной встречей
- Ведущий убеждается в этом перед началом
- Входные критерии могут быть такими:
  - Подготовлены чеклисты
  - Основополагающие документы вышли из инспекций с известным уровнем ошибок
  - За 10 минут пробной проверки был найден не более одного найденного дефекта
  - Документы грамматически проверены и пр...
- Выходные критерии — документы согласованы

ИТМО ВТ

Планирование инспекции

- Зависит от инспектируемых артефактов
  - Учитывать размер, сложность. Определяется «Check-rate»
  - Разбиение на одновременно обрабатываемые «куски»
- Зависит от участников
  - Убедиться, что все будут присутствовать
  - Обладают ли определенными знаниями и опытом
- Собираемые метрики (н-р: размеры, кол-во найденных дефектов, цена изменений, время на проверку)
- Разрабатывается расписание

ИТМО ВТ

Инспекция: обзор

- Включает:
  - Групповое обучение инспекторов
  - Назначение ролей инспекторам
  - Распространение материалов
- Ведущий представляет инспекторов

ИТМО ВТ

Инспекция: подготовка

- Сердце инспекции — здесь проводится само тестирование
- Инспекторы проверяют артефакты в соответствии с ролью
  - Отмечают время начала
  - Отмечают и записывают проблемы, фокусируясь на основных
  - Используют точно отведенное время (не больше, не меньше — check rate)
  - Классифицируют и считают проблемы (Major, minor, ?)

ИТМО ВТ

Инспекторы - роли

- Человек может фокусироваться только на двух проблемах одновременно
- Роли могут быть выбраны из следующих:
  - Чеклист — инспектор проверят по нему
  - Документы — инспектор проверяет целостность между несколькими документами
  - Фокус — поиск выделенных проблем
  - Перспектива — представить роль и будущее пользователя
  - Процедура — инспектор следует особой процедуре (.)
  - Сценарий — следование заданному сценарию (более специфично, чем предыдущий)
  - Стандарт — проверка на соответствие стандартам
  - Точка зрения — инспектирует с т.з. н-р пользователя

ИТМО ВТ

Инспекция - встреча, обсуждение

- Основное назначение — протоколирование результатов подготовки
  - Позволяет найти еще 10-20% проблем
  - Проблема: id, (Mmon), описание, документ, страница
- Ведущий:
  - Дипломатичен, краток, категоричен
  - Не позволяет развивать дискуссию, длительно предлагать решения
  - Принимает непопулярные решения (удалить к-л, остановить встречу)
- Ограничена по времени — не более 2-х часов
- Решение по артефакту — «принят», «на переработку», «отклонен»

ИТМО ВТ

Инспекция — переработка и выработка рекомендаций

- Если артефакт не принят - отправляется на переработку
  - Автором
  - Редактором
  - Другими работниками
- Инспекция не может завершиться, пока не удовлетворены выходные критерии
  - Проверяется ведущим
- Готовый артефакт передается в систему контроля версий для остальной команды

Статические методы. Обзор

	Сквозной контроль	**Технический Анализ**	**Инспекция**
Основное Предназначение	Поиск дефектов	Поиск дефектов	Поиск дефектов
Дополнительная цель	Обмен опытом	Принятие решений	Улучшение процесса
Подготовка	Обычно нет	Популяризация	Формальная подготовка
Ведущий	Автор	В зависимости от обстоятельств	Подготовленный модератор
Рекомендованный размер группы	2-7	>3	3-6
Формальная процедура	Обычно нет	Иногда	Всегда
Объем материалов	небольшой	От среднего до большого	небольшой
Сбор метрик	Обычно нет	Иногда	Всегда
Выходные данные	Неформалный отчет	Формальный отчет	Список дефектов, результаты метрик, формальный отчет

ИТМО ВТ

Средства статического анализа кода

- Большое количество (C - Lint, Java — FindBugs)
- Находят:
  - Неопределенное поведение (переменная не инициализирована)
  - Нарушение алгоритмов использования библиотеки (fopen без fclose)
  - Сценарии некорректного поведения
  - Переполнение буфера
  - Разрушение кроссплатформенности
  - Дефекты копи-пейста
  - …

ИТМО ВТ

Системное тестирование

5

ИТМО ВТ

Тестирование системы в целом

- Начинается после окончания интеграции
- Включает несколько фаз:
  - Системное тестирование — выполняется внутри организации-разработчика
  - Альфа- и Бета-тестирование — выполняется пользователем под контролем разработчика
  - Приемочное тестирование — выполняется пользователем.
    - Результат — платить или не платить.
- Методики практически одинаковые, различная строгость интерпретации результатов.

ИТМО ВТ

Системное тестирование

- Стандартная цель
- Фокус:
  - Функциональность системы с точки зрения пользователя
  - Нефункциональные характеристики
- Проводится валидация!
  - В основном метод черного ящика

ИТМО ВТ

Системное тестирование. Напоминание!

- Тестирование — это как научный эксперимент — необходимо повторение
- Автоматизация, скриптинг
- Результаты должны быть:
  - Записаны в журнал.
  - На каждый найденный дефект — зарегистрирован инцидент
  - Периодические отчеты для найденных ошибок

ИТМО ВТ

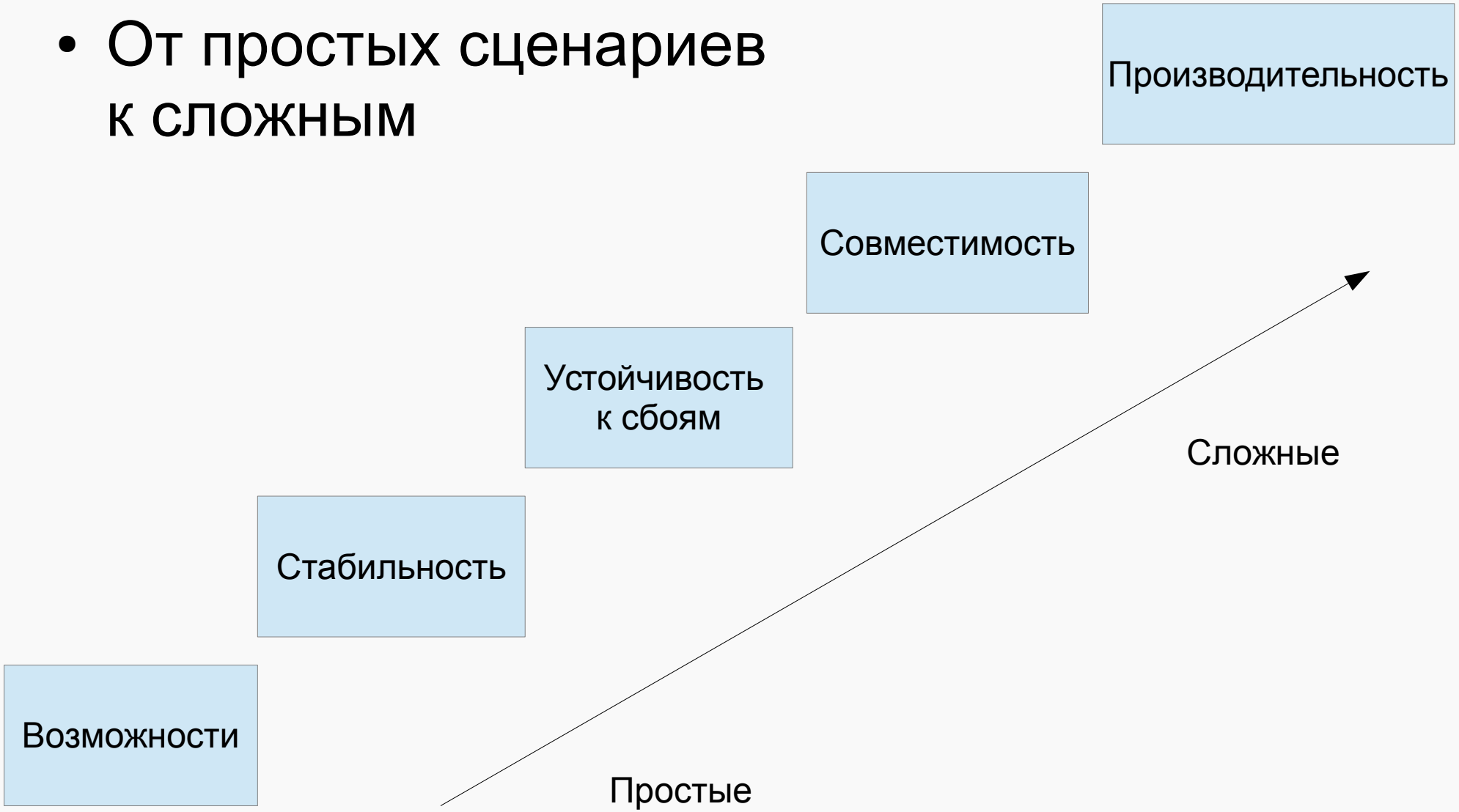
Контекст системного тестирования

- Тестовое окружение д.б. максимально близко к операционному
  - Оборудование
  - Операционная система
  - Сетевое окружение
  - В аппаратуре — частота, тайминги … и пр.
- Тестирование эмулирует реальные действия пользователей
  - Тестеры исполняют роли реальных пользователей
  - Скрипты исполняют реальные сценарии
- Система под управлением разработчиков

ИТМО ВТ

Где начать?

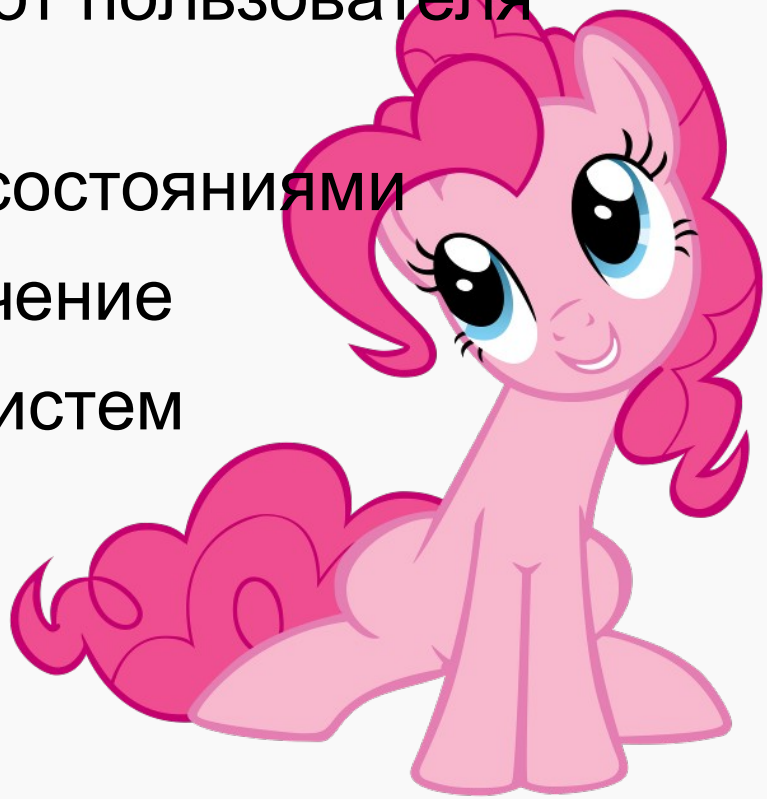
- От простых сценариев к сложным



ИТМО ВТ

Возможности

- Работают ли основные функции системы
  - Один пользователь
  - Одна транзакция
  - Корректный ввод информации от пользователя или из файлов, БД
  - Нормальные переходы между состояниями
  - «Среднее» аппаратное обеспечение
  - Положительные проверки подсистем безопасности



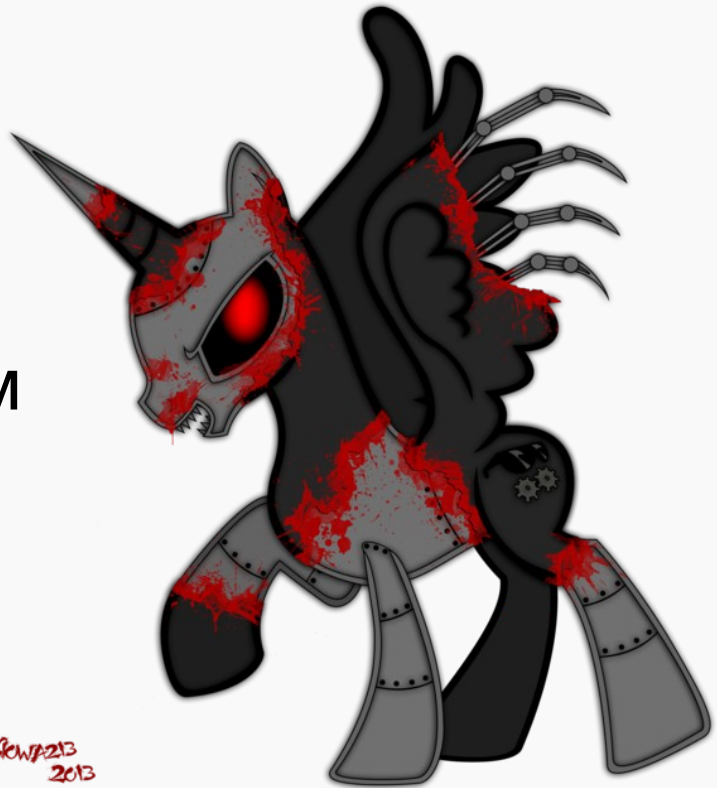
Стабильность

- Попробуем более реалистичную ситуацию
  - Все еще валидные входные данные и транзакции
  - Несколько пользователей одновременно
  - Реальные последовательности транзакций
  - Одновременно несколько транзакций от одного пользователя (если есть)
  - Запуск системы на длительный период
    - Утечки памяти, переполнения диска. ….

ИТМО ВТ

Устойчивость к сбоям

- Let's break it!
  - Пользователи вводят неправильные данные
  - Сбой сети, испорченные файлы
  - Неправильные переходы между состояниями
  - Выполнение операций в неправильном порядке
  - Аварийное отключение систем/подсистем
  - Намеренные ошибки подсистем безопасности
    - Плохие пароли, внешний взлом



ИТМО ВТ

Совместимость

- Проверяется функционирование в разных средах
  - Взаимодействие с локальными и удаленными системами
  - Проверка функционирования с различными версями
    - Библиотек
    - Браузеров
    - Операционных систем

Производительность - CARAT

- CARAT
  - Capacity — Нефункциональные возможности
  - Accuracy — Точность
  - Responce Time — Время ответа
  - Availability — Готовность
  - Throughput — Пропускная способность

ИТМО ВТ

Нефункциональные возможности

- Максимальное количество (пользователей, записей в БД, файлов, Кб, ГГц), поддерживаемое системой
- Одновременно, не нарушая других требований производительности

ИТМО ВТ

Точность

- Корректность:
  - Алгоритмов
  - Результатов
- Может быть критичной для определенного класса систем

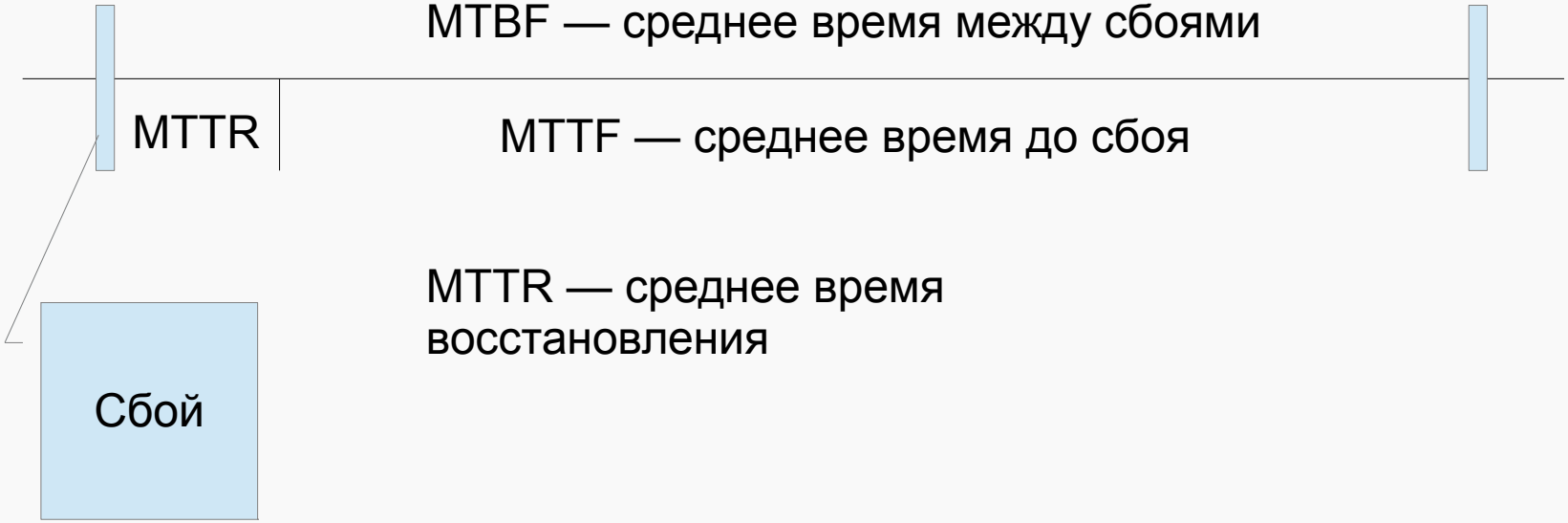
ИТМО ВТ

Время отклика

- Один из наиболее часто измеряемых и критичных критериев
  - Системы реального времени
- От «нажатия на кнопку» до «полной загрузки страницы»
- От подачи управляющего сигнала до наступления реакции
- Под минимальной, расчетной, пиковой нагрузкой

Готовность

- Коэф. готовности = (MTBF — MTTR)/ MTBF



- 0,99999 — сколько минут в год?

ИТМО ВТ

Пропускная способность

- Количество операций (транзакций) в секунду которые может поддерживать система
- Баланс с временем отклика
- Стресс-тестирование — последовательная нагрузка системы до момента неприемлемого времени отклика

ИТМО ВТ

Инструменты нагрузочного тестирования

- HP Load Runner
- LoadComplete
- IBM Rational Performance Tester
- Load UI Pro
- Apache Jmeter
- The Grinder
- Tsung

ИТМО ВТ

Нагрузочное тестирование с Jmeter

- Бесплатный, кроссплатформенный
- Интерфейс пользователя
  - Простой, строит графики
- Возможность создания распределенной нагрузки
- Эмуляция одновременной работы пользователей
- Снятие метрик
- Возможность разрабатывать плагины
- Планы тестирования в XML — просто контролировать версии

http://jmeter.apache.org

ИТМО ВТ

Jmeter и функциональное тестирование

- Может быть использован для записи запросов
- Jmeter → Функциональное тестирование → Нагрузочное тестирование
- Jmeter — не браузер!

ИТМО ВТ

Тестовый план

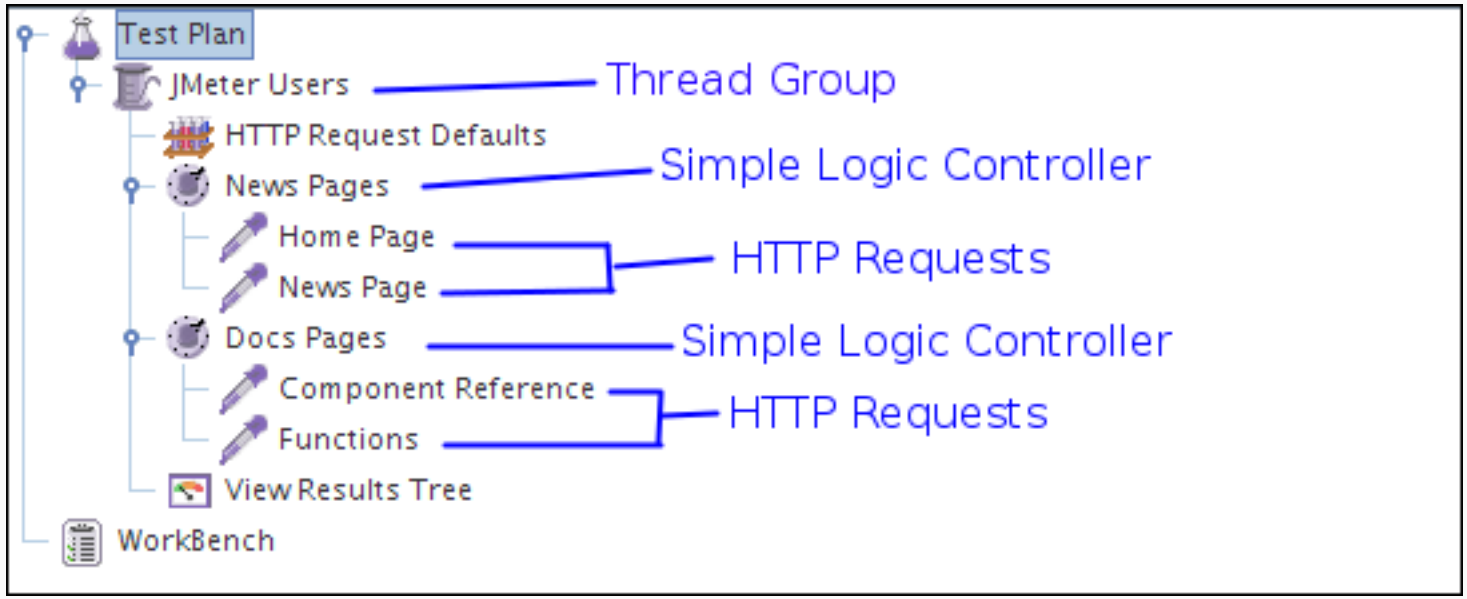
- Thread Group
  - Описывает пул пользователей для выполнения теста
  - Количество, возрастание и пр..
- Семплеры
  - Формируют запросы, генерируют результаты
  - Большой набор встроенных протоколов (TCP, HTTP, FTP, JDBC, SOAP, JMS, SMTP, …..)

http://jakarta.apache.org/jmeter/usermanual/component_reference.html

ИТМО ВТ

Тестовый план - 2

- Логические контроллеры
  - Определяют порядок вызова семплеров
  - Конструкции управления (if, loop, …)
  - Управление потоком



http://jakarta.apache.org/jmeter/usermanual/component_reference.html

ИТМО ВТ

Тестовый план - 3

- Слушатели
  - Получают ответы
  - Осуществляют доп. операции с результатами: просмотр, запись, чтение и др.
  - Не обрабатывают данные! (в командной строке, нужен GUI)
- Таймеры
  - Задержки между запросами
  - Постоянные, в соответствии с законами

http://jakarta.apache.org/jmeter/usermanual/component_reference.html

ИТМО ВТ

Тестовый план - 4

- Assertion
  - Проверяют результаты
- Элементы конфигурации
  - Сохраняют предустановленные значения для семплеров
- Препроцессоры
  - Изменяют семплеры в их контексте (HTML Link Parser)
- Постпроцессоры
  - Применяются ко всем семплерам в одном контексте

http://jakarta.apache.org/jmeter/usermanual/component_reference.html

ИТМО ВТ

Тестовый план — порядок выполнения

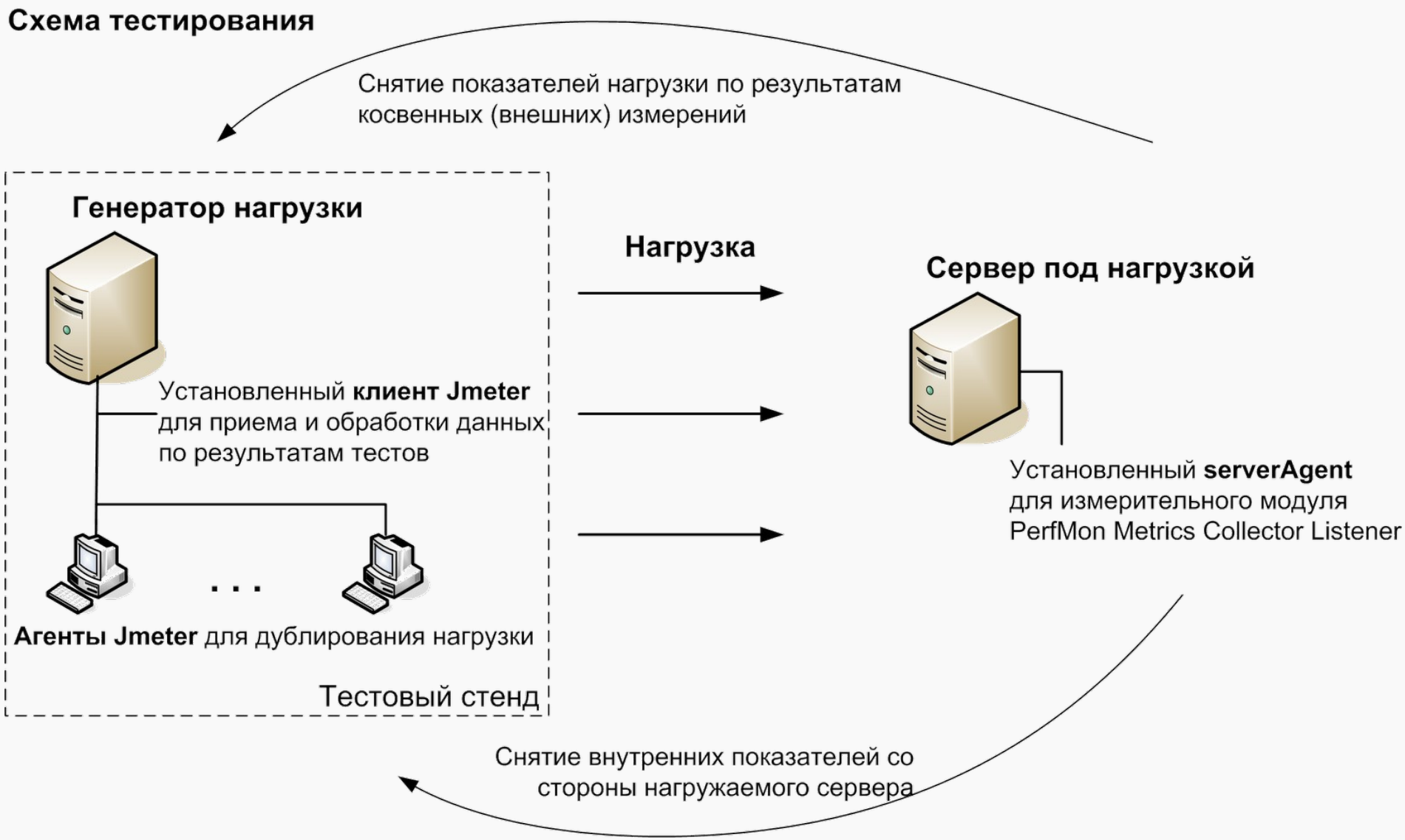
- Конфигурационные элементы
- Препроцессоры (Pre-Processors)
- Таймеры (Timers)
- Семплеры (Sampler)
- Постпроцессоры (Post-Processors)
- Assertions
- Слушатели (Listeners)

ИТМО ВТ

Jmeter — дополнительные возможности

- Автоматическая генерация CSV-файла
- Bandwidth Throttling - ограничение полосы пропускания
  - Статическое (CPS), динамическое (error rate)
- IP Spoofing — замена src_ip, для разделения клиентов
  - Проверка балансировщиков нагрузок
- Поддержка теста TPC-C
  - Стандартный тест на OLTP нагрузку

Jmeter — распределенное тестирование



http://forworktests.blogspot.ru/2013/03/jmeter.html

ИТМО ВТ

Альфа-тестирование

- Проводится небольшим количеством пользователей внутри организации, разработавшей ПО
  - Может проводиться до окончания системного тестирования
  - Ранние отзывы пользователей об использовании системы в рабочем окружении

ИТМО ВТ

Бета-тестирование

- Проводится выбранной группой пользователей в своем/рабочем окружении
- Windows Beta test — по всему миру
- Могут быть ошибки
- Может быть не реализован весь функционал
- Ранние отзывы для разработчиков
- Превью для пользователей

ИТМО ВТ

Альфа- и бета-тестирование

- Проводится неформально
  - Пользователи работают с тем функционалом, который им интересен
  - … отмечают то, что не работает
  - … доводят дефекты и сбои до разработчика
- Нет тестовых сценариев и планов
- Разработчики не гарантируют, что немедленно исправят проблемы. Исправления могут быть:
  - в финальном релизе;
  - в следующей бета-версии.

ИТМО ВТ

Приемочное тестирование

- Ключевой аспект — управляется пользователями
  - А не разработчиками
- Проверка системы на «соответствие предназначению и практическому использованию»
- Формальное
- Неформальное

ИТМО ВТ

Приемочное тестирование - 2

- Все ли требования удовлетворены
  - Функциональные
  - Нефункциональные
- Источник — документ «требования к системе» (RUP — SRS)
  - Подготовка может начинаться на ранних этапах
  - Может применяться статическое тестирование
- Ответственно подходить к выбору пользователей и обязательно их обучить!

ИТМО ВТ

Приемочное тестирование - 3

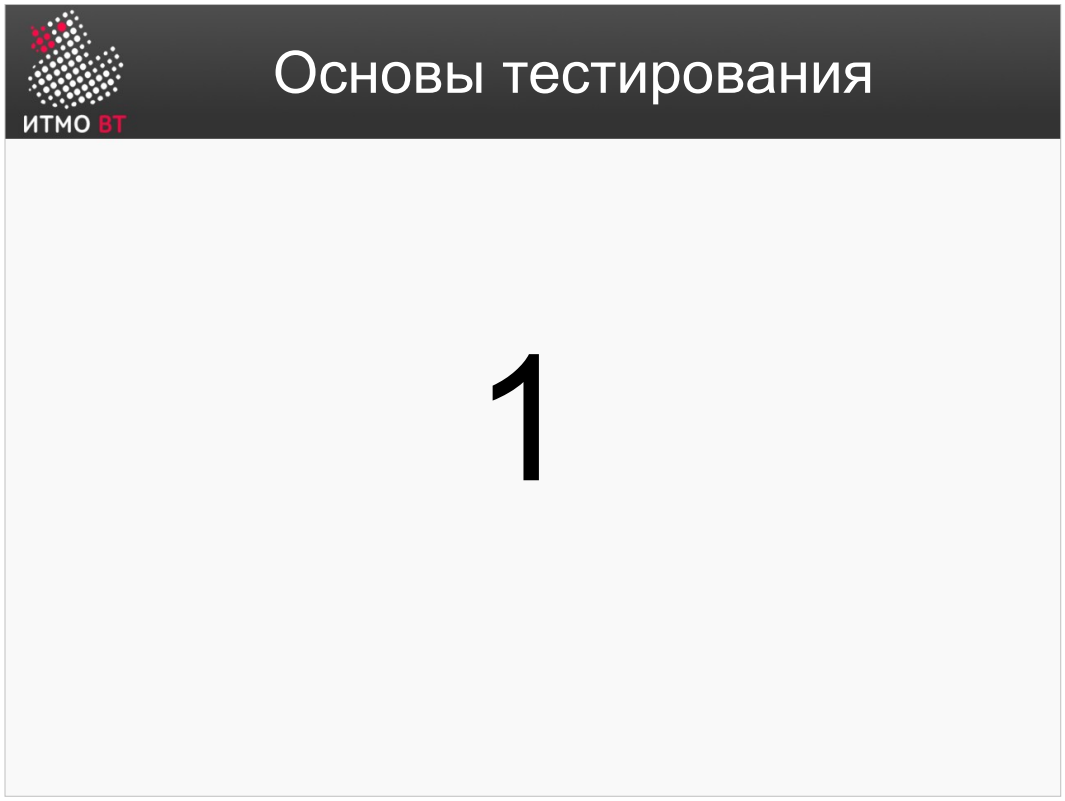
- Окружение должно быть полностью готово
  - Аппаратура и программное окружение
  - Может быть первый запуск в рабочем окружении
- Тесты проводятся и анализируются так же, как и в системном тестировании

Тестирование ПО

Клименков С.В.
2013-2014 уч. Год
v.1.05 от 24.04.2015

Литература

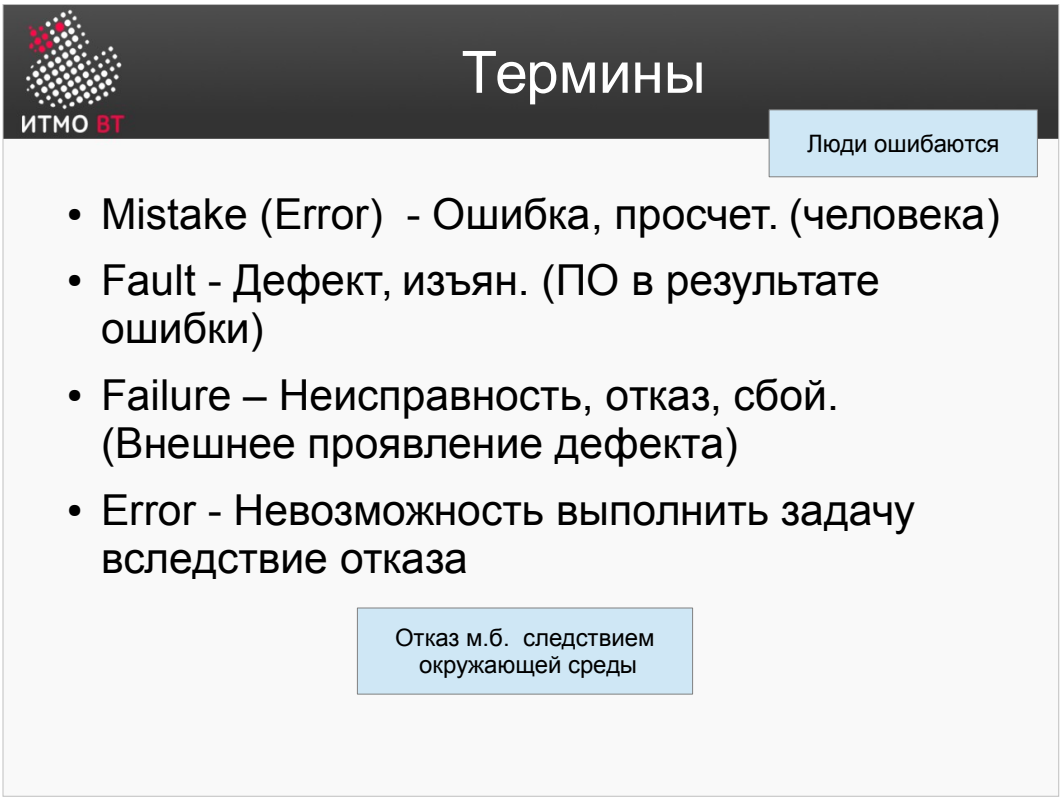
- Иан Соммервилл. Инженерия программного обеспечения
- Рекс Блэк «Advanced Software Testing»,
- Рекс Блэк Ключевые процессы тестирования.
- ISTQB - International Software Testing Qualifications Board
  - www.istqb.org
  - www.rstqb.org
- P.Ammann, J.Offutt
  Introduction to Software Testing
  - www.cs.gmu.edu/~offutt/softwaretest/


ИТМО ВТ
Основы тестирования
1

Системы с программным обеспечением

- Бизнес-системы
- Аппаратура с ПО
- Потребительские товары
- Военные, космические системы
- Информационно-управляющие системы
- ...

Сильное давление со стороны бизнеса и гибких методологий



Терминология

Помеха, дефект, ошибка, недочет, просчет, качество, риск

Software Fault : A static defect in the software
Software Failure : External, incorrect behavior with respect to the requirements or other description of the expected behavior
Software Error : An incorrect internal state that is the manifestation of some fault

Mistake - Ошибка. Человеческое деяние, которое в конечном итоге привело к получению неверного результата. В широком смысле – непреднамеренное отклонение от истины или правил. Оригинал: A human action that produces an incorrect result.

Fault - Дефект, изъян. Неверный шаг (или процесс, или определение данных) в компьютерной программе. Следствие ошибки, потенциальная причина неисправности. Оригинал: An incorrect step, process, or data definition in a computer program. The outgrowth of the mistake, potentially leads to failure.

Failure - Неисправность. Неправильный результат. Собственно, результат дефекта. Оригинал: An incorrect result. The result of the fault (e.g. a crash).

Error - Невозможность выполнить задачу (или получить верный результат) вследствие того, что где-то случилась ошибка, которая привела к дефекту, который вызвал неисправность, которая привела к невозможности сделать то, чего мы тут намеревались. Оригинал: A failure to complete a task, usually involving a premature termination.

Объяснение в ISTQB:

Ошибка (error или mistake) — это акт программиста (человека, human being), который приводит к появлению дефекта (defect) в программе. И этот дефект там живёт, может быть долго, и всё хорошо, но в какой-то момент случается сбой (fault или failure).

Сбой — это видимое проявление дефекта. А дефект — это результат ошибки, совершённой программистом.

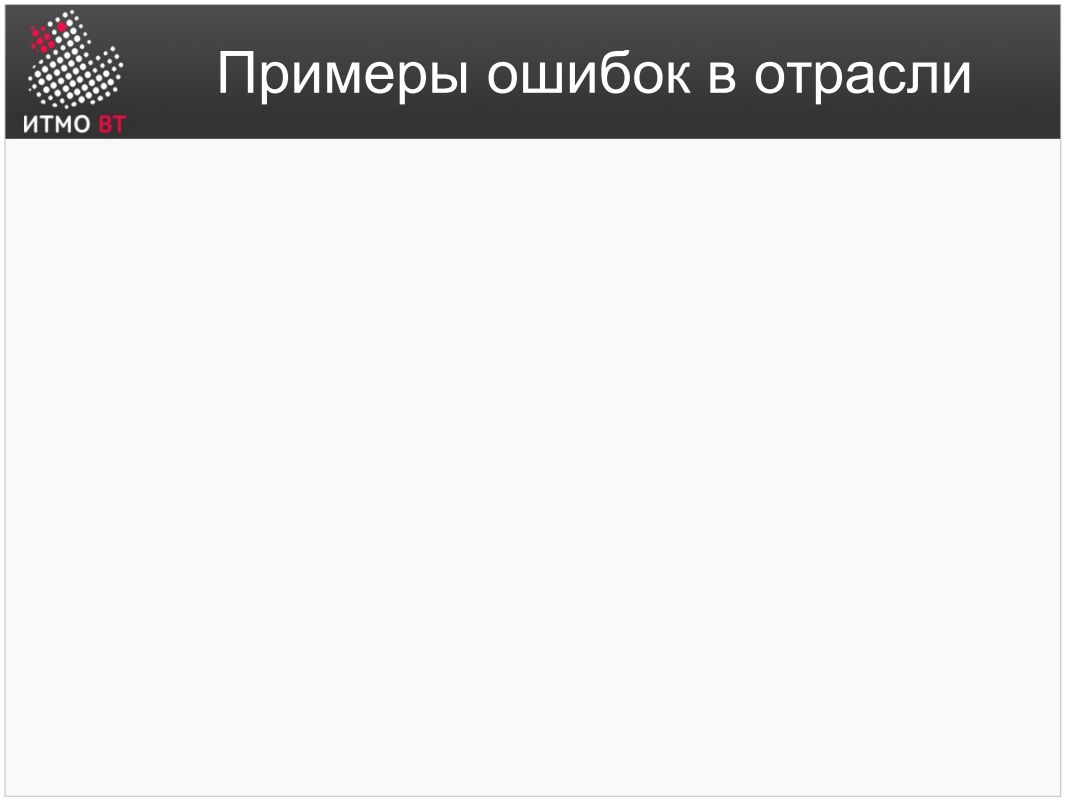
Пример программы

```
public  int countPositive (int [ ] data) {
   int count = 0;
   for (int i = 1; i < data.length; i++) {
      if (data [ i ] > 0)
         count++;
   }
   return count;
}
```

Сколько здесь дефектов
и к чему они могут привести?

BUG

- Используется неформально
- Может обозначать
  - Дефект (fault)
  - Отказ (failure)
  - Невозможностью выполнить задачу (error)
  - Что то другое или ничего не обозначать


ИТМО ВТ
Примеры ошибок в отрасли

Уровни восприятия тестирования

ИТМО ВТ

- Уровень 0 – тестирование == отладка
  - Не отличает некорректное поведение и ошибки в программе
  - Не учитывает требования надежности и безопасности
- Уровень 1 - предназначение – показать корректность ПО

HW engineer

  - Невозможно доказать
  - Что значит “ошибок нет”?
  - Нет формальных правил

“Тестирование может показать наличие дефектов, а не их отстутствие”
Edgar Dijkstra

Уровни восприятия тестирования

ИТМО ВТ

- Уровень 2 - Демонстрация ошибок

SW компании

  - Конфликт разработчиков и тестировщиков
- Уровень 3 - Тестирование может показать наличие ошибок
  - Используя ПО мы подвержены рискам
  - Риск – последствия незначительные
  - Риск – последствия катастрофические
  - Тестировщики и разработчики **совместно** снижают риски

«просветленные» SW компании

Тестирование и качество ПО

- Уровень 4 — Тестирование - это возможный способ оценки качества программного обеспечения в терминах найденных дефектов
  - Функциональное
  - Нефункциональное :надежность, практичность, эффективность, сопровождаемость и переносимость
  -
- ISO 9126 «Информационная технология. Оценка программного продукта»

Традиционные производства

Тестирование и качество

- Другие способы оценки качества
  - Разработка стандартов
  - Обучение
  - Анализ дефектов

Цели тестирования (ISTQB)

- Цели тестирования:
  - Обнаружение дефектов
  - Повышение уверенности в уровне качества
  - Предоставление информации для принятия решений
  - Предотвращение дефектов

Цели тестирования

- Увеличение приемлемого уровня пользовательского доверия в том, что программа функционирует корректно во всех необходимых обстоятельствах
  - Корректное поведение
  - Уровень доверия
  - Необходимые обстоятельства - требование реального окружения

Корректное поведение

- Необходимо определение
  - Из требований
  - Спецификаций, описаний, …
  - Зависит от уровня тестирования

ИТМО ВТ

Уровень доверия

- Наглядность
- Уровень остаточного обнаружения дефектов
  - Число дефектов обнаруженных тестом или набором тестов
  - Число дефектов обнаруженных в заданное время

> «Меньше 10-ти критических дефектов найдено за последние 7 дней»

- Требования к надежности
  - Сложно показать без испытаний, т. е. работающего ПО

> Среднее время между отказами не должно быть меньше 5000 часов

Реальное окружение

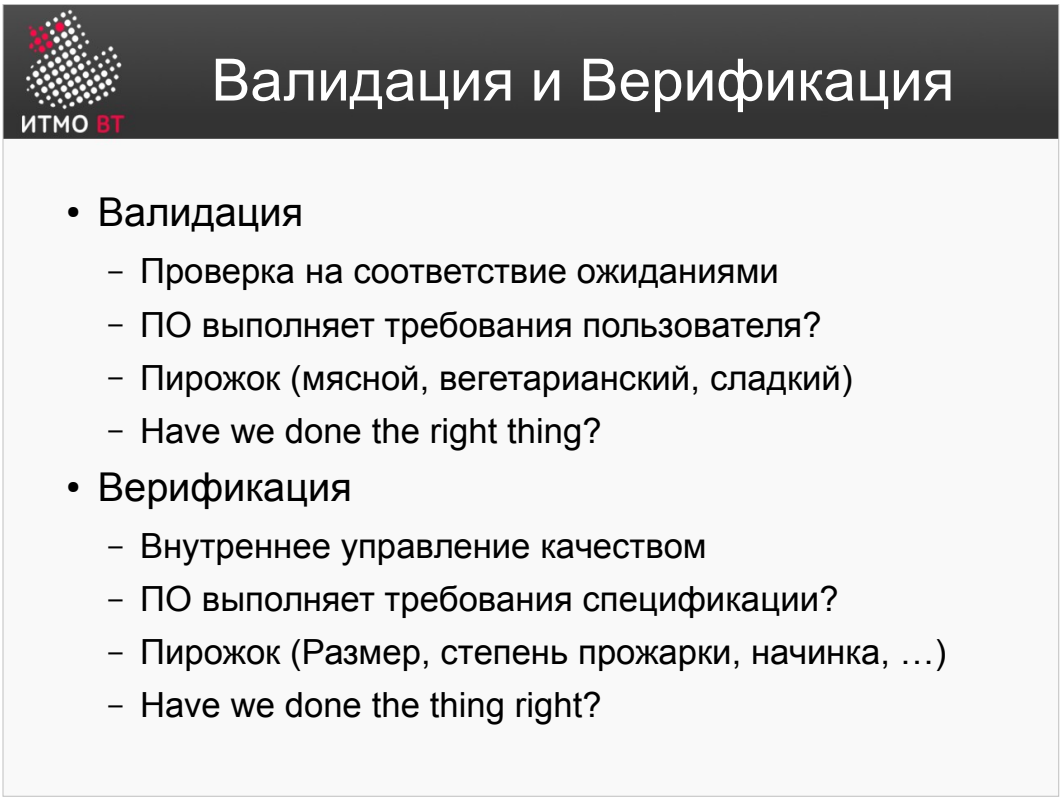
- Реалистичное количество данных - таких же как в целевой системе
  - В университете 5000 студентов, небольшой рост
  - Необходим тест на 5000, 6000, 7000 студентов, но не на 100000
- Реалистичный набор, комбинация входных данных

Полное тестовое покрытие

- public long multiply (int A, int B)
  - Как протестировать?
  - Сколько протребует памяти?
  - Сколько будет выполнятся на 3Ггц ЦПУ?

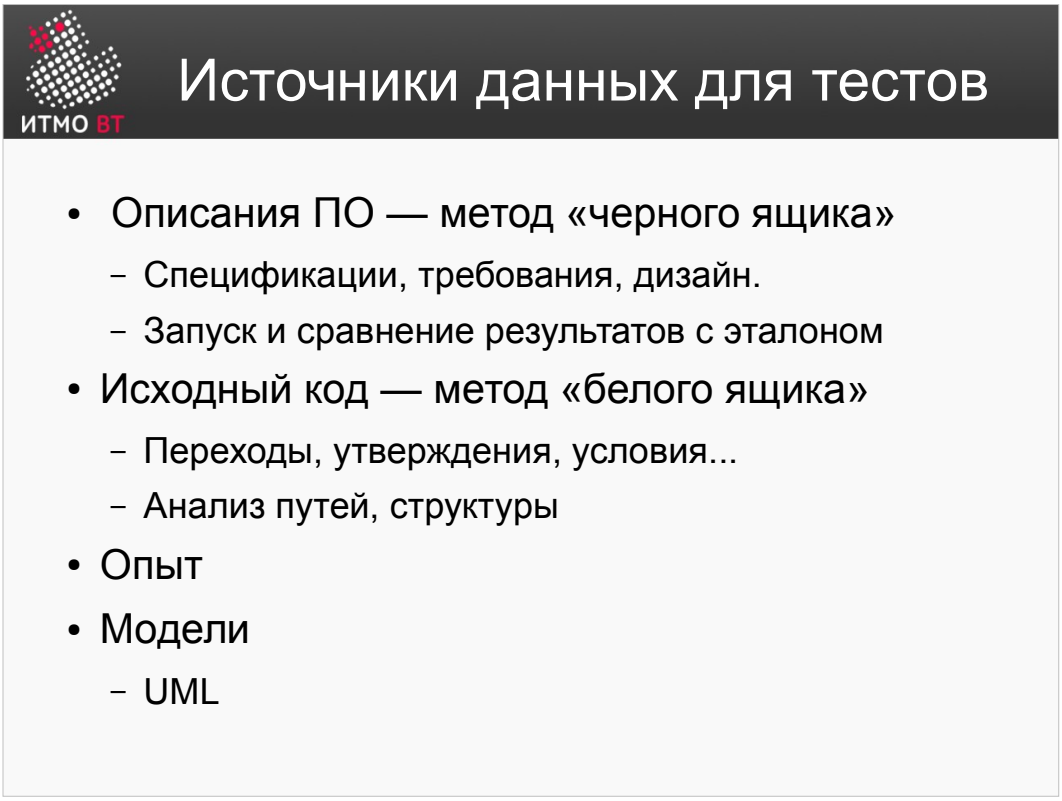
Статическое и динамическое тестирование

- Статическое (рецензирование)
  - Не включает выполнения кода
  - Ручное, автоматизированное
  - Неформальное, сквозной контроль, инспекция
- Динамическое
  - Запуск модулей, групп модулей, всей системы
  - После появления первого кода (а иногда перед!)



валидация (validation): Доказанное объективными результатами исследования подтверждение того, что требования для ожидаемого конкретного использования приложения были выполнены. [ISO 9000

верификация (verification): Доказанное объективными результатами исследования подтверждение того, что определенные требования были выполнены. [ISO 9000]



Классическим является разделение методов тестирования на методы черного и белого ящиков. Методы черного ящика (включающие в себя методы разработки тестов на основе спецификаций и на основе опыта) – это способ определить и выбрать тестовые условия или сценарии для компонента или системы (как функциональные, так и не функциональные), на основе анализа базиса тестирования и опыта разработчиков, тестировщиков и пользователей, без отсылки к внутренней структуре компонента или системы. Методы белого ящика (также называемые структурными, или основанными на структуре) основываются на анализе структуры компонента или системы. Оба этих метода могут быть объединены с методом создания тестов на основе опыта, чтобы использовать опыт разработчиков, тестировщиков и пользователей для определения того, что должно быть протестировано. Некоторые методы могут быть однозначно отнесены к определенной категории, другие же сочетают в себе несколько категорий.

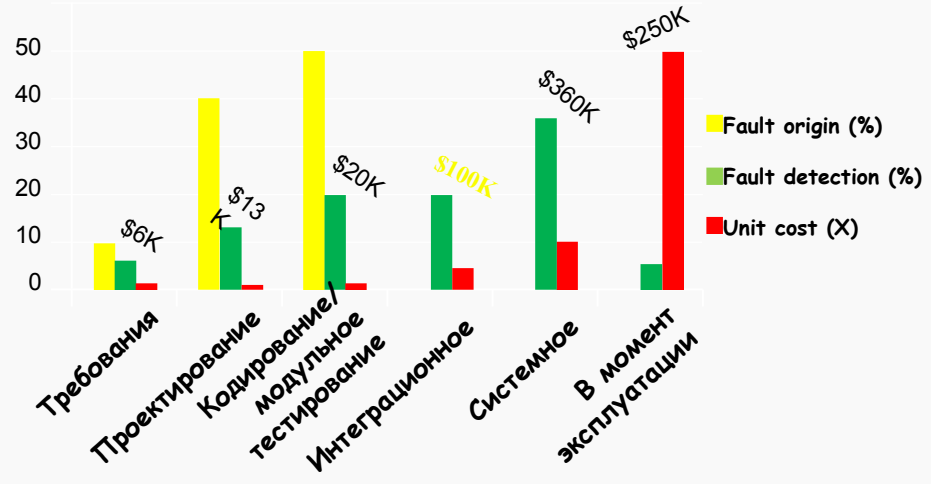
Деятельность и роли в тестировании

ИТМО ВТ

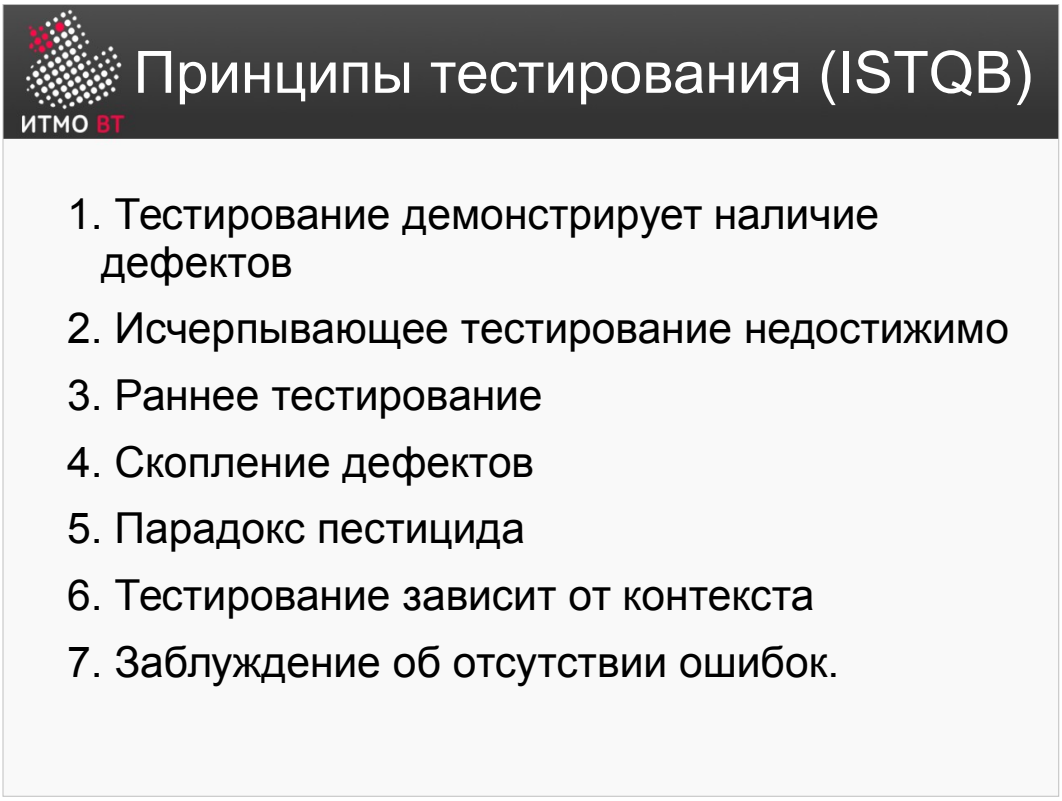
- Проектирование тестов
  - На основании формальных критериев
  - На основании знаний предметной области, опыта и экспертизы
- Автоматизация тестов
  - Знание средств, скриптов
- Исполнение тестов
  - Нет специальных требований к квалификации
- Анализ результатов
  - Знания предметной области

1a.	Design	Design test values to satisfy engineering goals
	Criteria	Requires knowledge of discrete math, programming and testing
1b.	Design	Design test values from domain knowledge and intuition
	Human	Requires knowledge of domain, UI, testing
2.	Automation	Embed test values into executable scripts Requires knowledge of scripting
3.	Execution	Run tests on the software and record the results Requires very little knowledge
4.	Evaluation	Evaluate results of testing, report to developers Requires domain knowledge

Цена позднего тестирования

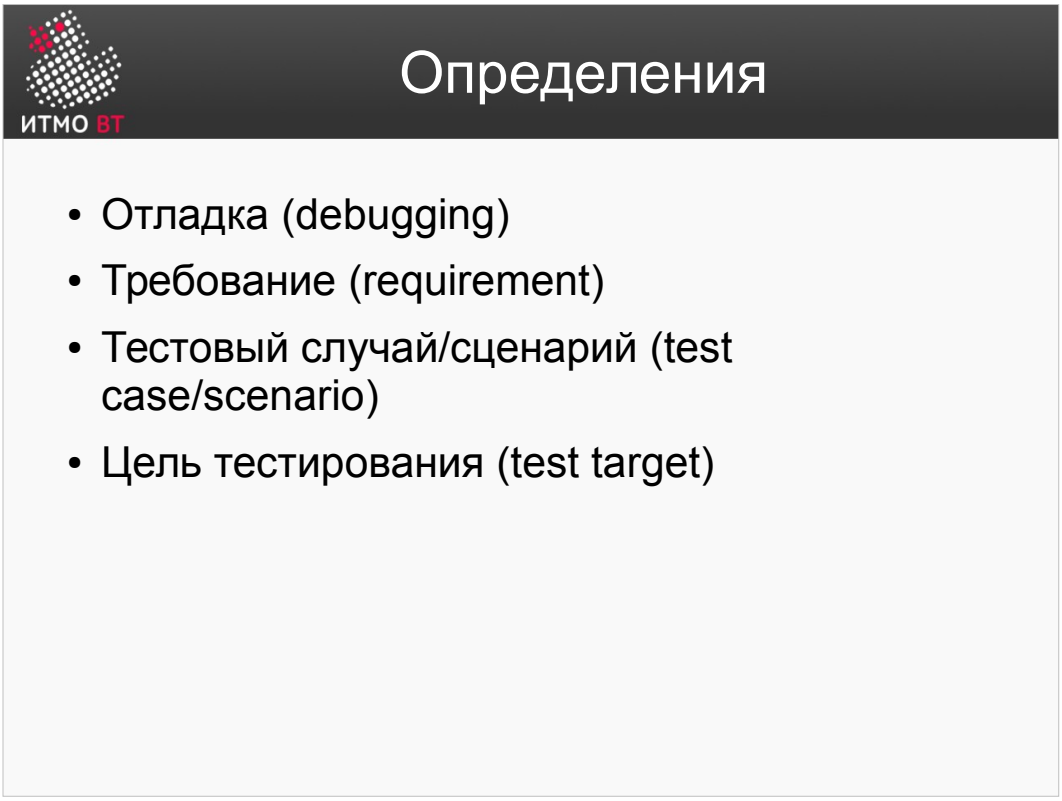


Software Engineering Institute; Carnegie Mellon University; Handbook CMU/SEI-96-HB-002

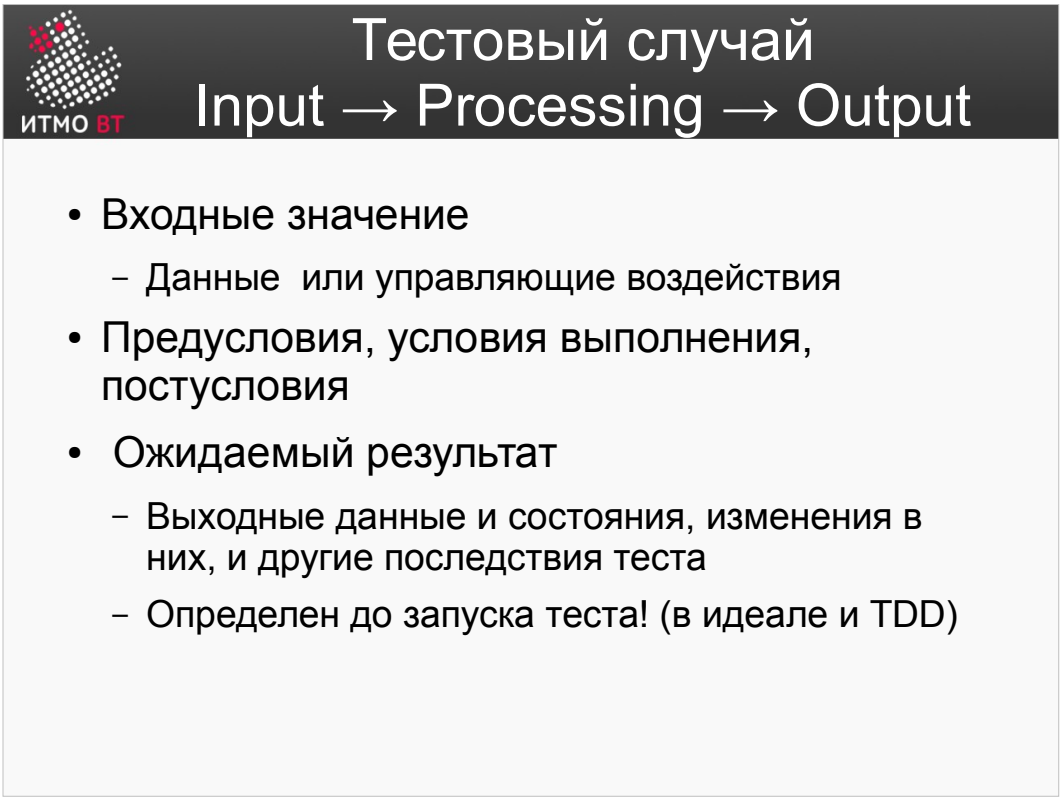


Принцип 1 – Тестирование демонстрирует наличие дефектов
Тестирование может показать, что дефекты присутствуют, но не может доказать, что их нет. Тестирование снижает вероятность наличия дефектов, находящихся в программном обеспечении, но, даже если дефекты не были обнаружены, это не доказывает его корректности.
Принцип 2 – Исчерпывающее тестирование недостижимо
Полное тестирование с использованием всех комбинаций вводов и предусловий физически невыполнимо, за исключением тривиальных случаев. Вместо исчерпывающего тестирования должны использоваться анализ рисков и расстановка приоритетов, чтобы более точно сфокусировать усилия по тестированию.
Принцип 3 – Раннее тестирование
Чтобы найти дефекты как можно раньше, активности по тестированию должны быть начаты как можно раньше в жизненном цикле разработки программного обеспечения или системы, и должны быть сфокусированы на определенных целях.
Принцип 4 – Скопление дефектов
Усилия тестирования должны быть сосредоточены пропорционально ожидаемой, а позже реальной плотности дефектов по модулям. Как правило, большая часть дефектов, обнаруженных при тестировании или повлекших за собой основное количество сбоев системы, содержится в небольшом количестве модулей.
Принцип 5 – Парадокс пестицида
Если одни и те же тесты будут прогоняться много раз, в конечном счете этот набор тестовых сценариев больше не будет находить новых дефектов. Чтобы преодолеть этот “парадокс пестицида”, тестовые сценарии должны регулярно рецензироваться и корректироваться, новые тесты должны быть разносторонними, чтобы охватить все компоненты программного обеспечения, или системы, и найти как можно больше дефектов.
Принцип 6 – Тестирование зависит от контекста
Тестирование выполняется по-разному в зависимости от контекста. Например, программное обеспечение, в котором критически важна безопасность, тестируется иначе, чем сайт электронной коммерции.


ИТМО ВТ
Основы тестирования
2



•Отладка (debugging): Процесс поиска, анализа, и устранения причин отказов в ПО.
•Требование (requirement): Условия или возможности, необходимые пользователю для решения определенных задач или достижения определенных целей, которые должны быть достигнуты для выполнения контракта, стандартов, спецификации, или других формальных документов. [IEEE 610]
•тестовый сценарий (test case): Набор входных значений, предусловий выполнения, ожидаемых результатов и постусловий выполнения, разработанный для определенной цели или тестового условия, таких как выполнения определенного пути программы или же для проверки соответствия определенному требованию.
•цель тестирования (test target) Набор критериев выхода.



Тестовый сценарий состоит из набора входных значений, предусловий выполнения, ожидаемых результатов и постусловий, определяемых для покрытия определенных тестовых условий (или тестового условия) или целей (цели) тестирования. Содержание спецификаций проектирования тестов (включая тестовые условия) и спецификаций тестовых сценариев описывается в стандарте «Документация при тестировании программ» (IEEE STD 829-1998

Ожидаемые результаты должны создаваться как часть спецификаций тестовых сценариев и включать в себя выходные данные, изменения в данных и состояниях, и любые иные последствия теста. Если ожидаемые результаты не были определены, правдоподобные, но ошибочные результаты могут быть приняты за корректные. В идеальных условиях ожидаемые результаты должны быть определены до момента выполнения теста.)

Тестовый случай (2)

- Повторяемый, автоматизируемый
- Учитывает состояния (если есть)
  - Переходы между состояниями
    - Правильные: корректный результат
    - Неправильные : корректные сообщения об ошибках
- В российской официальной терминологии используется термин сценарий

ИТМО ВТ

Тестовый сценарий

- Последовательность случаев
  - Типичное использование системы

№	Начальное состояние	Ввод	Действие системы	Вывод	Конечное состояние
1	Готов	Пользователь вставляет карточку	Успешное чтение карточки	Приглашение “введите pin”	Ожидание pin-кода
2	Ожидание pin-кода	Вводим **верный** pin-код	Проверка pin-кода	Приглашение к выбору транзакции	Ожидание выбора транзацкии
3	Ожидание выбора транзацкии	Выбор выдачи 5000 рублей	Проверка баланса, возможности выдачи	Деньги	Выдача денег
4	Выдача денег	Пользователь берет деньги и карточку	Завершение выдачи	Благодарность за использование	Готов

ИТМО ВТ

Тестовые сценарии (2)

- Должны обрабатывать
  - Корректное поведение и вариант ошибки

№	Начальное состояние	Ввод	Действие системы	Вывод	Конечное состояние
1	Готов	Пользователь вставляет карточку	Успешное чтение карточки	Приглашение “введите pin”	Ожидание pin-кода
2	Ожидание pin-кода	Вводим **неверный** pin-код	Проверка pin-кода	Сообщение об ошибке	Ожидание pin-кода
3	Ожидание pin-кода	Вводим **неверный** pin-код	Проверка pin-кода	Сообщение об ошибке	Ожидание pin-кода
4	Ожидание pin-кода	Вводим **неверный!!!** pin-код	Проверка pin-кода, блокировка карточки	Сообщение об ошибке и блокировка	Готов

Тестовые сценарии (3)

- Определение корректного поведения в:
  - Требованиях (системное, приемочное тестирование)
  - Архитектуре (интеграционное тестирование)
  - Проектных документах (модульное тестирование)
- Тестовые сценарии
  - Можно взять из документации к проекту: Use-case → Test-case

Сколько тестов?

- Требуется баланс
  - Много тестов → больше покрытие → качество выше
  - Меньше тестов → выше скорость разработки → быстрее выход на рынок
- Необходимо выбрать специфические значения для тестирования
  - Нельзя же тестировать вечность!
  - Полное покрытие недостижимо

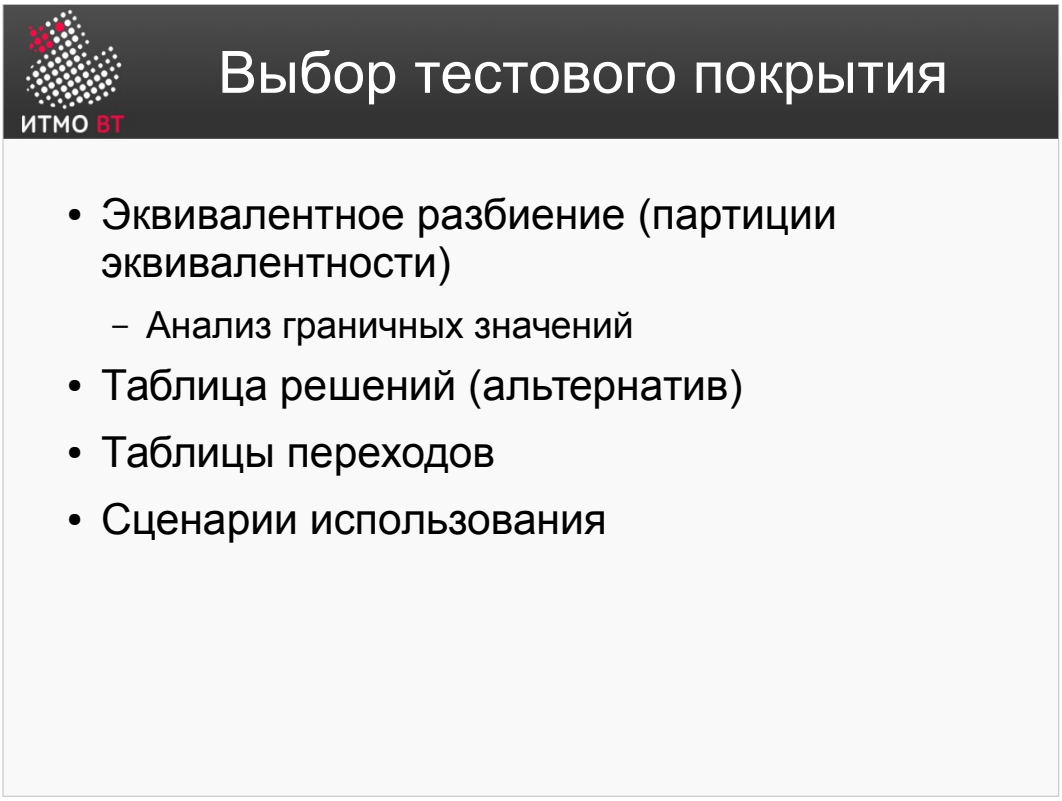


таблица решений (decision table): Таблица, отражающая комбинации входных данных и/или причин с соответствующими выходными данными и/или действиям (следствиям), которая может быть использована для проектирования тестовых сценариев.

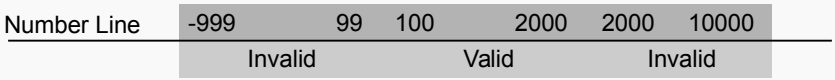
Анализ эквивалентности

- Набор входных данных на которых результат является эквивалентным или подобным
  - Например, если вывод системы одинаковый — находимся внутри эквивалентной области
  - Дополнительно может использоваться анализ граничных значений
- Банкомат:
  - Купюры по $100
  - За раз максимум $2000
  - За день $10000
  - Комиссия 1% но не меньше $5

Анализ эквивалентности (2)

Сумма к снятию	Сумма к списанию	Результат	Класс результата
-1	-	нет	невалидный
1..99,101..199, ...	-	нет	невалидный
100,200,..,500	105,205, …, 505	выдача	валидный
600,700,...,2000	606,707, …, 2020	выдача	валидный
2100,2200...	-	нет	невалидный
2000+1..99	-	нет	невалидный
2000+100	2020+105	Выдача в два этапа	валидный
2000+2000+2000+ 2000+2000	10100	Выдача в 5 этапов	валидный

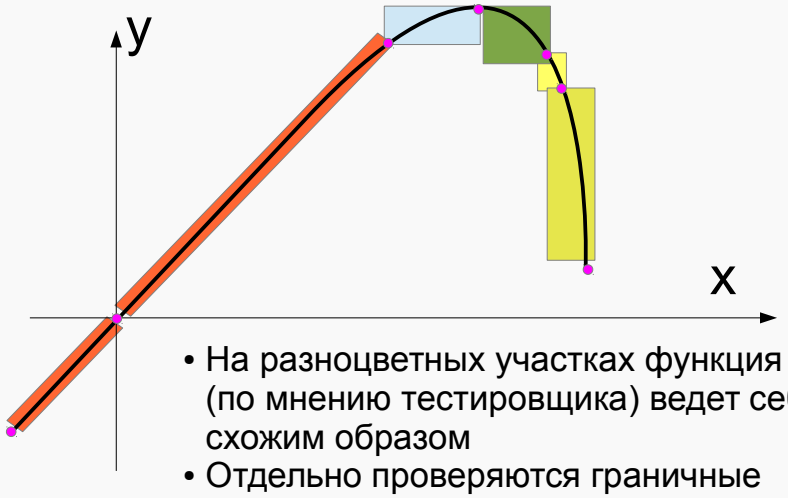
Граничные значения для параметра выдачи средств



ИТМО ВТ

Анализ эквивалентности

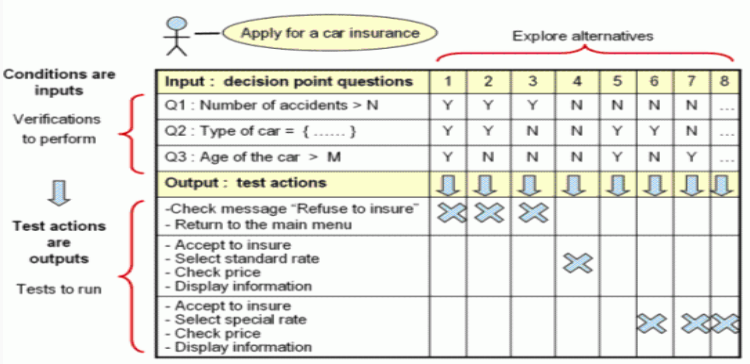
Зависимость удовлетворения от количества съеденного холодца



- На разноцветных участках функция (по мнению тестировщика) ведет себя схожим образом
- Отдельно проверяются граничные значения

Таблица решений

- Используется в системах со сложной логикой, описание конечного автомата
- Может включать большой набор условий (обычно true-false), и действий



Input : decision point questions	1	2	3	4	5	6	7	8
Q1 : Number of accidents > N	Y	Y	Y	N	N	N	N	...
Q2 : Type of car = { ..... }	Y	Y	N	N	Y	Y	N	...
Q3 : Age of the car > M	Y	N	N	N	Y	N	Y	...
Output : test actions	⇩	⇩	⇩	⇩	⇩	⇩	⇩	⇩
-Check message "Refuse to insure" - Return to the main menu	X	X	X					
- Accept to insure - Select standard rate - Check price - Display information				X				
- Accept to insure - Select special rate - Check price - Display information						X	X	X

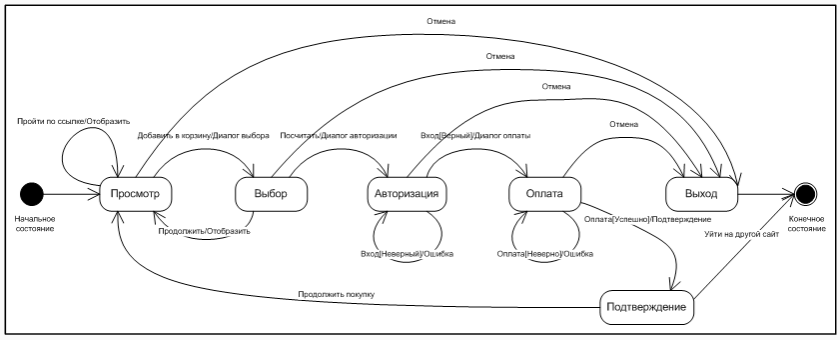
http://software-testing.ru/library/testing/functional-testing/129--functional-tester-ibm-rational-

Полезная статья
http://inrecolan.ru/blog/viewpost/330

ИТМО ВТ

Таблицы переходов

- Позволяют выбрать состояния и их комбинации, которые можно опустить
  - Рекс Блэк «Advanced Software Testing».
  - http://inrecolan.ru/blog/viewpost/361



http://inrecolan.ru/blog/viewpost/360

http://inrecolan.ru/blog/viewpost/361

Таблицы переходов (2)

- Прокрыть определенные строки тестами
- while (есть тесты с «непокрытыми» строками)
  - Выбрать состояния «не определено»
  - Попытаться покрыть тестом

Состояния		События/Условия		Текущее состояние	Событие/Условие	Действие	Новое состояние
				Просмотр	Пройти по ссылке	Отобразить	Просмотр
		Пройти по ссылке		Просмотр	Добавить в корзину	Диалог выбора	Выбор
		Добавить в корзину		**Просмотр**	**Продолжить**	**Не определено**	**Не определено**
		Продолжить		**Просмотр**	**Выписать**	**Не определено**	**Не определено**
Просмотр		Выписать		**Просмотр**	**Вход [неверный]**	**Не определено**	**Не определено**
Выбор		Вход [неверный]		**Просмотр**	**Вход [верный]**	**Не определено**	**Не определено**
Авторизация	X	Вход [верный]	=	**Просмотр**	**Оплата [неверно]**	**Не определено**	**Не определено**
Оплата		Оплата [неверно]		**Просмотр**	**Оплата [успешно]**	**Не определено**	**Не определено**
Подтверждение		Оплата [успешно]		**Просмотр**	**Отмена**	**Нет действия**	**Выйти**
Выйти		Отмена		**Просмотр**	**Продолжить покупку**	**Не определено**	**Не определено**
		Продолжить покупку		**Просмотр**	**Уйти на другой сайт**	**Не определено**	**Не определено**
		Уйти на другой сайт		**Выбор**	**Пройти по ссылке**	**Не определено**	**Не определено**
				{53 строки, сгенерированных по шаблону, не показаны}			
				Выйти	**Уйти на другой сайт**	**Не определено**	**Не определено**

http://inrecolan.ru/blog/viewpost/361

Сценарии использования (функциональное тестирование)

Прецедент: ViewInformationAboutTheRoute
ID: 2
Краткое описание: Пользователь просматривает информацию о маршруте
Главные актёры: Клиент (или любой другой пользователь)
Второстепенные актёры: нет
Предусловия: Пользователю выведен список маршрутов. Пользователь может быть не авторизован в системе
Основной поток: Прецедент начинается когда Пользователь выбирает конкретный маршрут Система отображает подробную информацию по выбранному маршруту

- Выбираем сценарий
- Проверяем его

Автоматизация тестов

- Регрессионное тестирование
- Повторение тестового сценария
- Приемочное тестирование
- Сокращение ручного труда?
- Проверка одного приложения в разных окружениях


ИТМО ВТ
Модульное и интеграционное тестирование
3

ИТМО ВТ

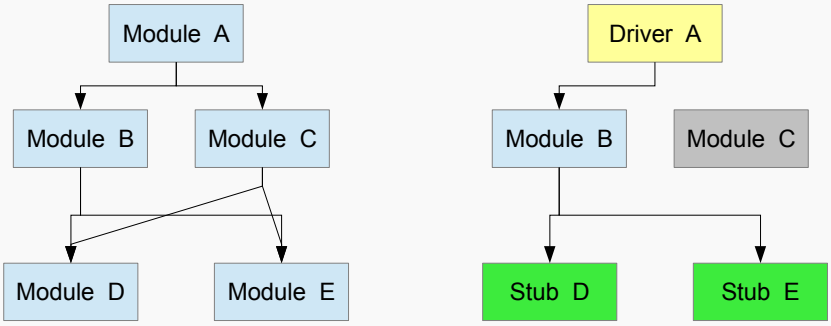
Модуль

- Модульное (компонентное) тестирование - тестирование отдельных компонентов программного обеспечения [IEEE 610].
  - Метод или класс
  - Программный модуль
- Модули описаны в дизайне
- Необходимо изолировать модуль из системы

ИТМО ВТ

Изолирование модулей

- Драйвер вместо вызываемой
- Заглушка вместо подчиненной



Заглушка

- Эмулирует поведение подчиненной программы
  - Подпрограмма, функция, процедура
  - Аппаратное прерывание, передача данных
- Интерфейс совпадает, внутренность нет
  - Предопределенные ответы для заданных аргументов, исключения
  - Постоянная генерация прерываний
- Используются вместо реальной программы
  - Компилируется и линкуется

Заглушка (2)

- Простая
  - Нельзя тестировать заглушку!
- Обычно 1 строка кода
- Возможна дополнительная логика
  - 50 раз возвращать 42, на 51 — IndexOutOfBounds
- Может читать значения из текстового файла
- Возможна настройка драйвером перед выполнением

Драйверы

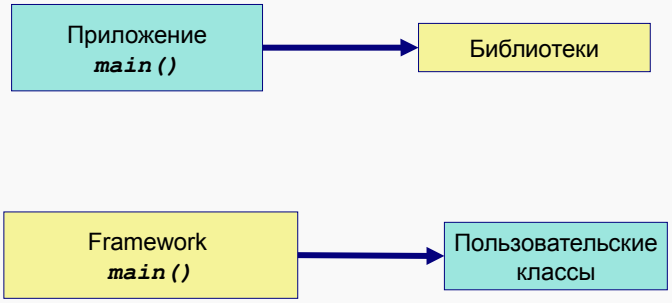
- Эмулирует вызываемый компонент
- Обычно уже более сложная программа
  - Устанавливает окружение
  - Подготавливает входные данные
- Дополнительно может:
  - Запускать серию тестов
  - Настраивать заглушки
  - Формировать журнал результатов

Test Fail

- Для анализа результата тестов программисту необходимо:
  - Входные и выходные значения
  - Значения измененных переменных
  - Пройденные пути, принятые решения
  - Симптомы сбоя
- Где ошибка в тесте или в ПО?

Фреймворки для модульного тестирования

- XXXunit — много разных!
- Фреймворк vs Библиотека



Junit 4

- JUnit фреймворк обеспечивает:
  - Аннотации для маркировки метода как теста @Test
  - Аннотации для маркировки действий до и после теста @Before, @After, @BeforeClass, @AfterClass
  - Методы для проверки (assertion)
  - UI, журнал тестов...

Пример

```
import org.junit.*;
import static org.junit.Assert.*;
public class MoneyTest {
   private Whale whale;
   @Before public void setUp() {
         whale = new Whale();
          whale.setLocation(”Где-то высоко”);
   }
   @Test public void testDown() {
         whale.fallDown();
         assertEquals(“Кит не упал”,
                        “на земле”,
                       whale.getLocation());
   }
}
```

Дополнительные возможности

```
@Test (timeout=10)
 public void longLoop() {
    assertEquals(PI.computeNNumber(1E10) ,3);
}
@Test (expected=IllegalArgumentException.class)
public void testSqrt() {
    Math.Sqrt(-5);
}
```

@RunWith(value=Parameterized.class) и @Parameters

@Ignore и автобоксинг

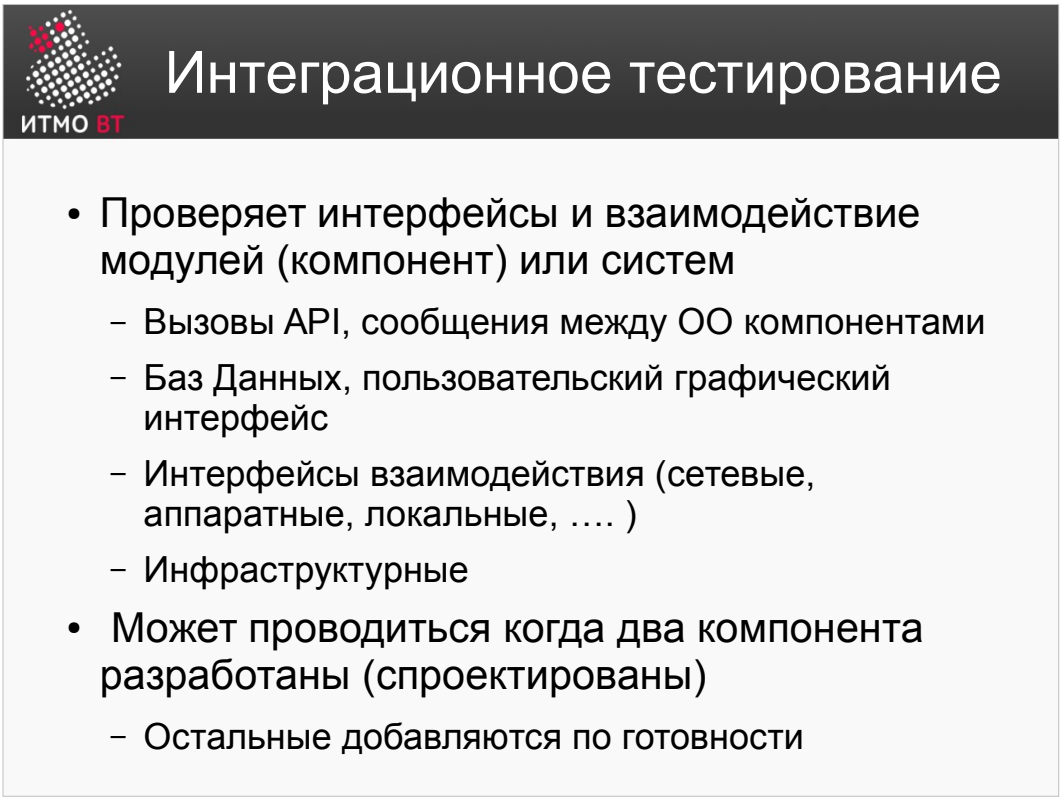
```
long sum(long x, long y) { return x + y; }

@Ignore("Евгений, помогите!")
@Test
public void add() {
    assertEquals(4, program.sum(2, 2));
}
```

expected: <4> but was: <4>

Автобоксинг

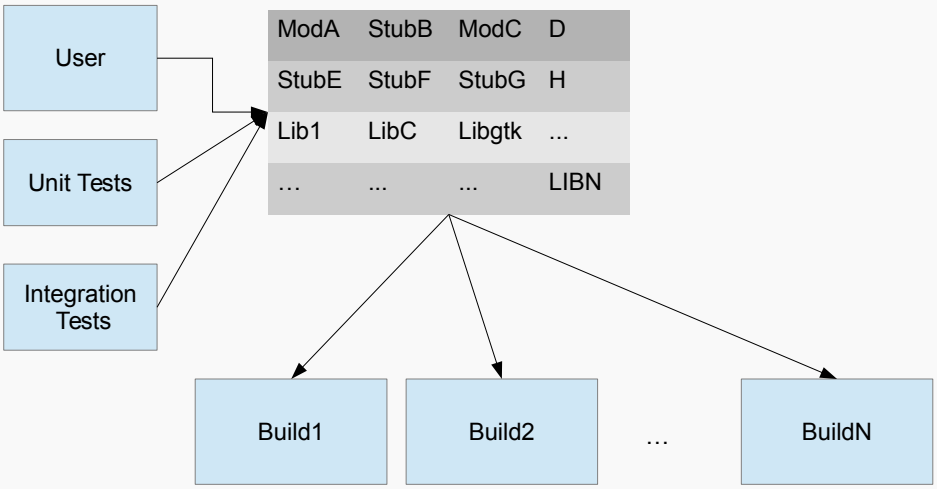
- В JUnit4 assertEquals не существует для примитивных типов
  - В Тесте 4 is упакуется в Integer, sum возвращает long
- Сообщение об ошибке значит:
  - expected int 4, but got long 4
- Надо исправить 4 to a 4L
  - assertEquals(4L, program.sum(2, 2));



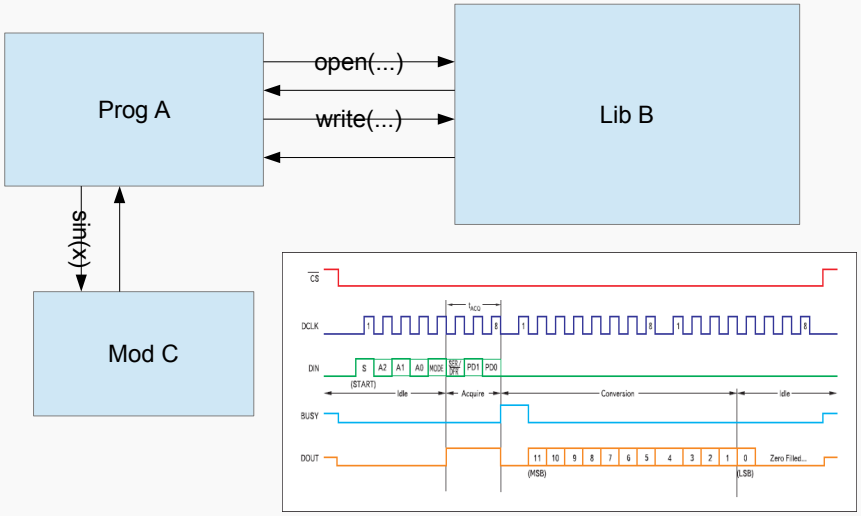
1. Компонентное интеграционное тестирование проверяет взаимодействие между программными компонентами и производится после компонентного тестирования
2. Системное интеграционное тестирование проверяет взаимодействие между системами или между аппаратным обеспечением и может быть выполнено после системного тестирования. В этом случае, разработчики могут управлять только одной стороной интерфейса. Однако, это может рассматриваться как риск. Бизнес-процессы могут включать последовательность систем; могут быть важны кроссплатформенные различия.

ИТМО ВТ

Интеграция

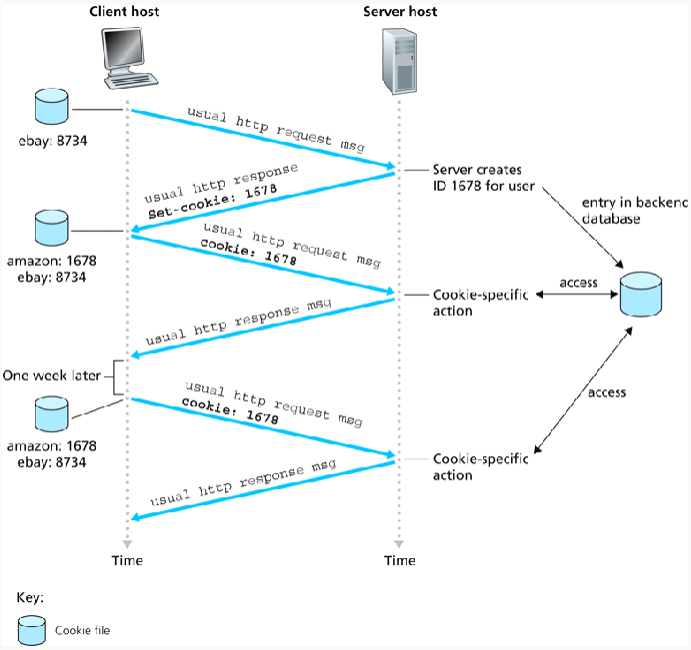


Вызовы и сообщения

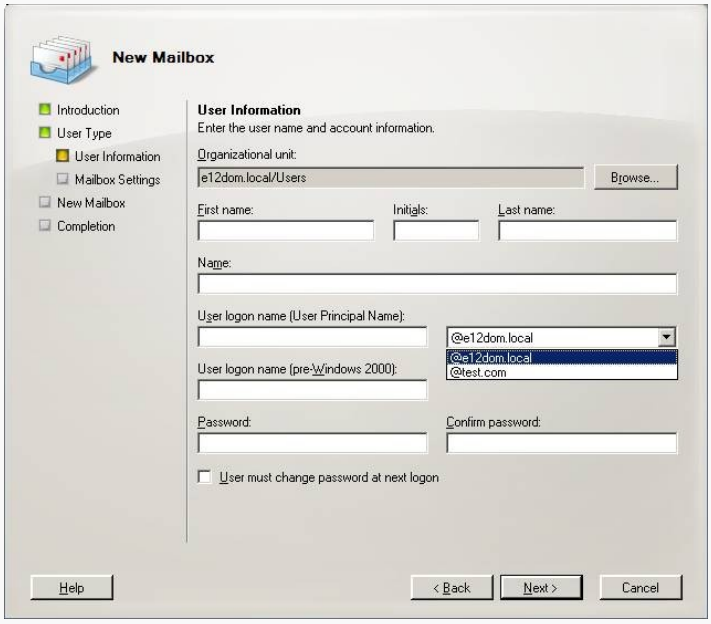


ИТМО ВТ

Сетевое взаимодействие



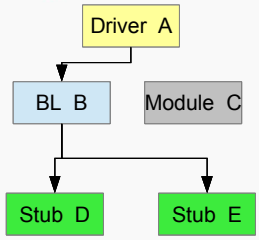
Пользовательские интерфейсы

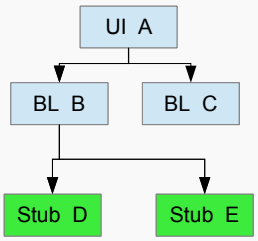


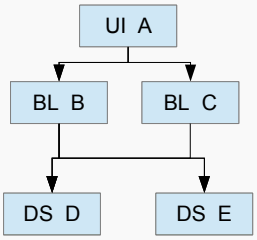
Стратегии интеграции

- Больше объем интеграции — больше сложность
  - Для каждого интерфейса должен быть разработан короткий тест план
- Выбор в зависимости от архитектуры ПО
- Последовательность имеет значение
- Можно тестировать нефункциональные характеристики

Сверху вниз

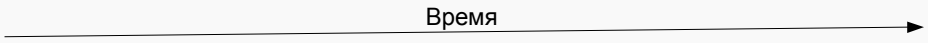






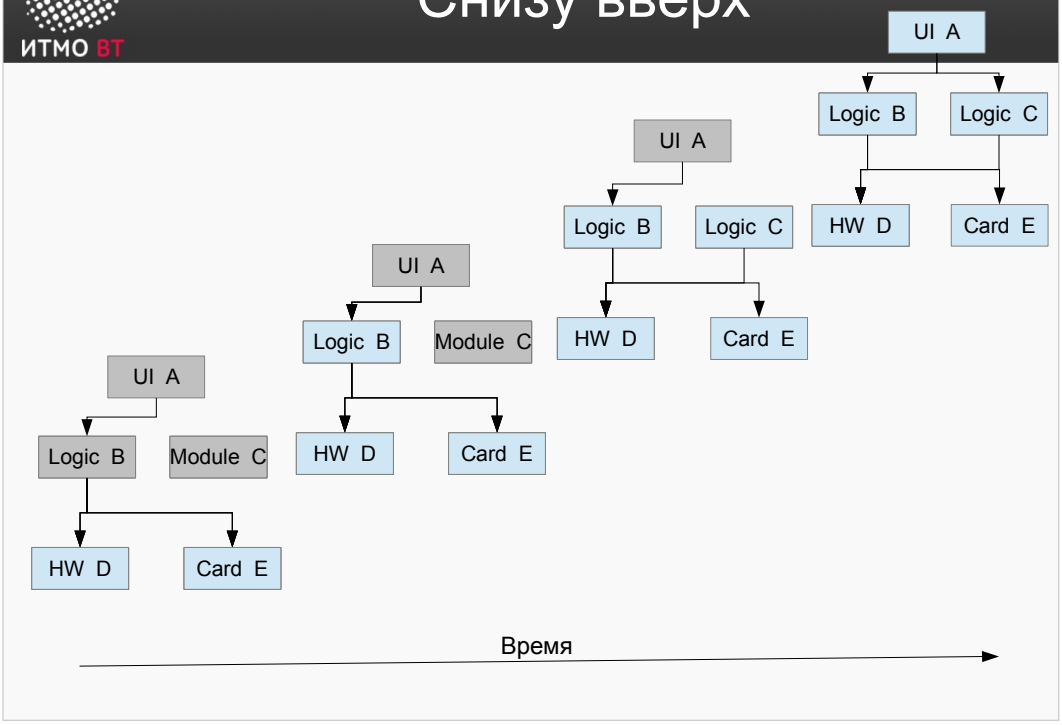
BL — бизнес логика
UI — интерфейс пользователя
DS — уровень хранения

\+ Быстро появляется «осязаемое» приложение
\- Большое количество заглушек



ИТМО ВТ

Снизу вверх



Другие

- Функциональная (end to end) — по одной функции
  - Собрали 1 сценарий UI-Логика-БД, добавили еще один такой-же
- Ядро (backbone)
  - Экран, клавиатура, мышь работают с минимальными функционалом
  - Добавить цвета на экран, колесо прокрутки …
- Большой взрыв (big bang)
  - Собрать все вместе и молиться

Основные принципы

- Интегрировать в первую очередь наиболее сложные компоненты
  - Снизит риски
  - Больше времени для исправления ошибок
- Выбирать порядок интеграции с учетом порядка разработки
- Использовать регрессионное тестирование после каждого этапа интеграции
- Автоматизировать тесты

Функциональное тестирование

- На базе сценариев использования
- Ручное/автоматическое
- На готовой системе, в рамках модульного и интеграционного
- Проверяются функции системы начиная с интерфейса пользователя
- Средства автоматизации
  - Открытые: Selenium, Sahi, Watir
  - Коммерческие: от HP, Rational (IBM) ….

Selenium

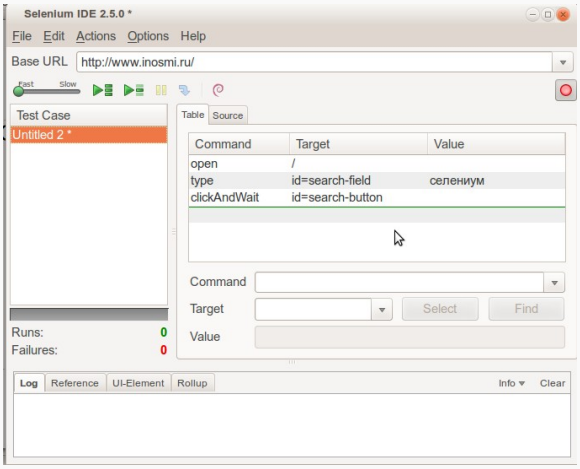
- Набор средств автоматизации
  - IDE
  - Selenium Server / WebDriver
  - Grid
- Тестирование web-приложений
- Кросс- браузерный
- Разработка сценариев на многих языках
  - java, php, python, c#
- Встроенные конструкции assert
- Встроенный механизм логирования ошибок

Selenium IDE

- Интегрированная среда исполнения и разработки тестов
- Разработано как расширение Firefox
- Запись тестов в виде Java, Ruby, HTML
- Добавление assert и команд

ИТМО ВТ

Selenium IDE



Команды

- click или clickAndWait — ссылки, переключатели, radio-кнопки
- type — ввод значений
- select — выбор значений из списка
- open — открывает страницу
- assert***
- wait*** - ожидание события
- verify *** - проверка

Assertion & Verification

- Предназначены для проверки содержимого элемента UI
  - Элемент присутствует?
  - Есть ли необходимый текст на странице?
- Если verification неуспешна — тест продолжается
- Если неуспешна assertion — тест останавливается

Добавление

- Добавляется во время записи теста щелчком правой кнопки мыши на элементе
  - verifyTextPresent
  - verifyTitle
  - verifyElementPresent
  - verifyValue
- Assertion - аналогично

Синхронизация или WaitFor Command

- waitForPageLoad(timeout) загрузка страницы ошибка по таймауту
- waitForAlert
- waitForTable — полная загрузка таблицы
- waitForTitle — загрузка заголовка
- Другие команды синхронизации

Другие команды

- store — сохранение значений в переменной
- echo — запись значения в лог selenium
  - Можно использовать ${var}

Запись скрипта в виде кода (фрагмент)

```
public class HelloTest {
  private WebDriver driver;
  private String baseUrl;
  private boolean acceptNextAlert = true;
  private StringBuffer verificationErrors = new StringBuffer();

  @Before
  public void setUp() throws Exception {
    driver = new FirefoxDriver();
    baseUrl = "http://www.inosmi.ru/";
    driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);
  }

  @Test
  public void testHello() throws Exception {
    driver.get(baseUrl + "/");
    driver.findElement(By.id("search-field")).clear();
    driver.findElement(By.id("search-field")).sendKeys("селениум");
    driver.findElement(By.id("search-button")).click();
  }
```

Ограничения SeleniumIDE

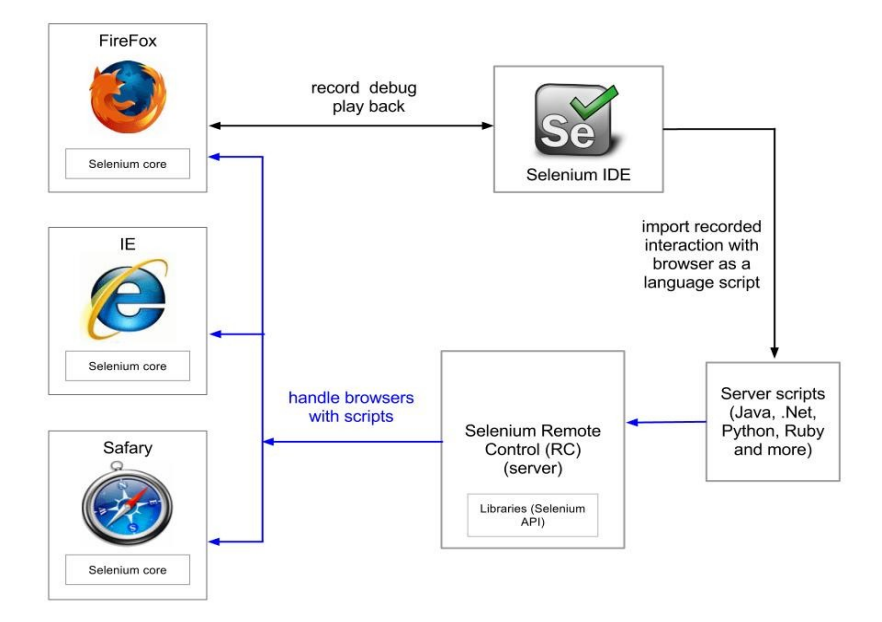
- Запускает тесты только в Firefox
- Слабо развитое управление логикой теста (циклы, условия, ….)
- Запускает только свои сценарии
- Сложно использовать с динамическим содержимым

Selenium Server

- Может быть использован для кросс-браузерного тестирования
- Сервер для тестирования разработан на java
- Работает как прокси для web-запросов совместно с Webdriver для каждого браузера
- Используется совместно с многими языками программирования
- Разработка — плагины к NetBeans и Eclipse
- Запуск тестов — maven и ant

ИТМО ВТ

Принцип работы



Процесс разработки

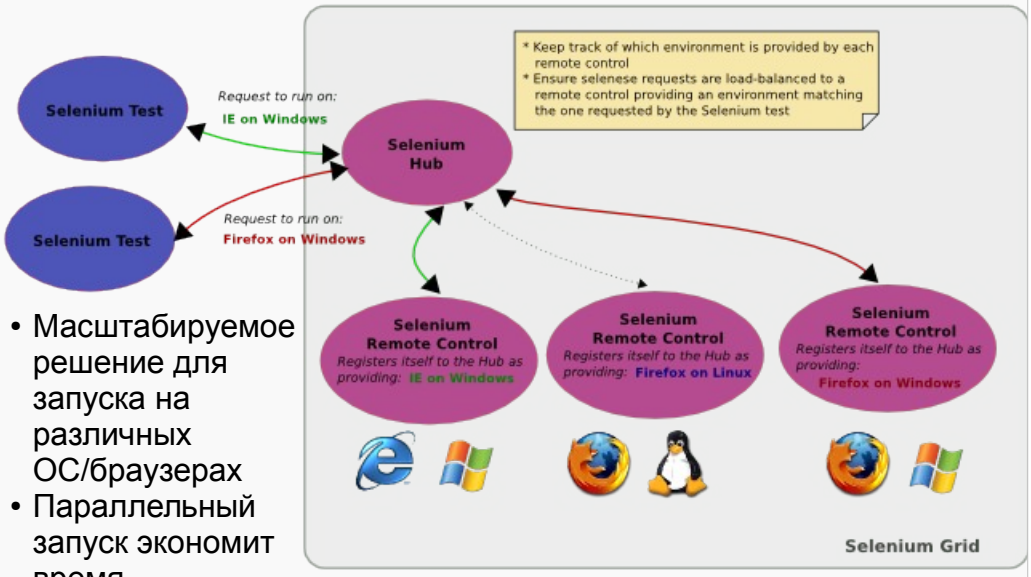
- Записать сценарий в виде java кода
- Создать проект в IDE и добавить необходимые jar файлы (например)
  - junit-4.8.1.jar
  - selenium-java-client-driver.jar
  - selenium-server.jar
  - testng-5.12.jars
- Скопировать сценарий в IDE
- Запустить

Xpath локатор

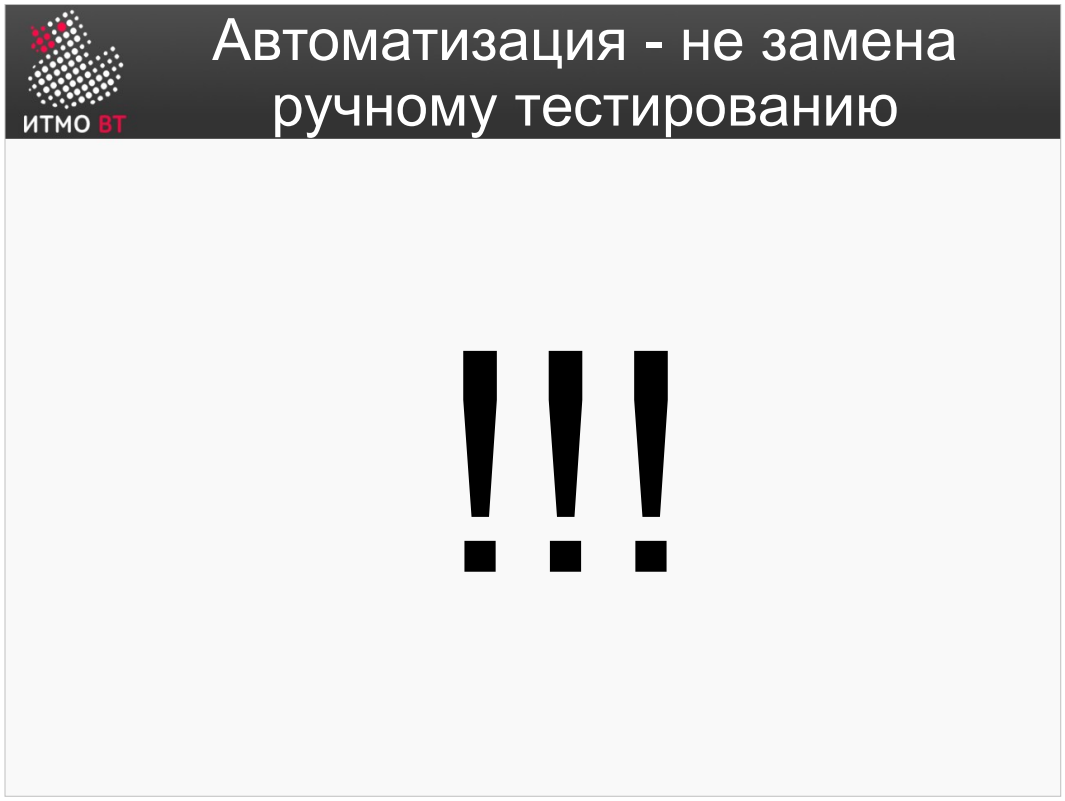
- В современных серверах приложений, порталах нельзя привязываться к компонентам по ID — он динамический
  - Используются классы стилей через xpath
- http://ru.wikipedia.org/wiki/XPath
- //div[contains(@class, 'article-heading')]

Selenium Grid

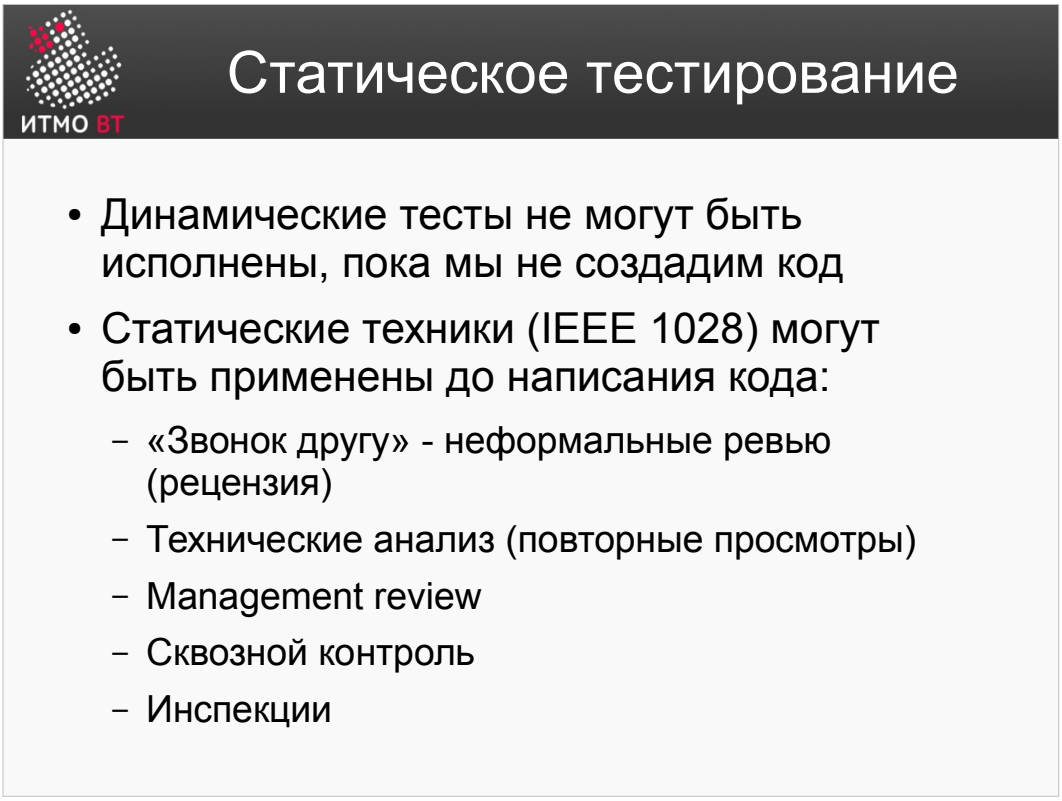
Selenium Grid : Requesting a Specific Environment



- Масштабируемое решение для запуска на различных ОС/браузерах
- Параллельный запуск экономит время

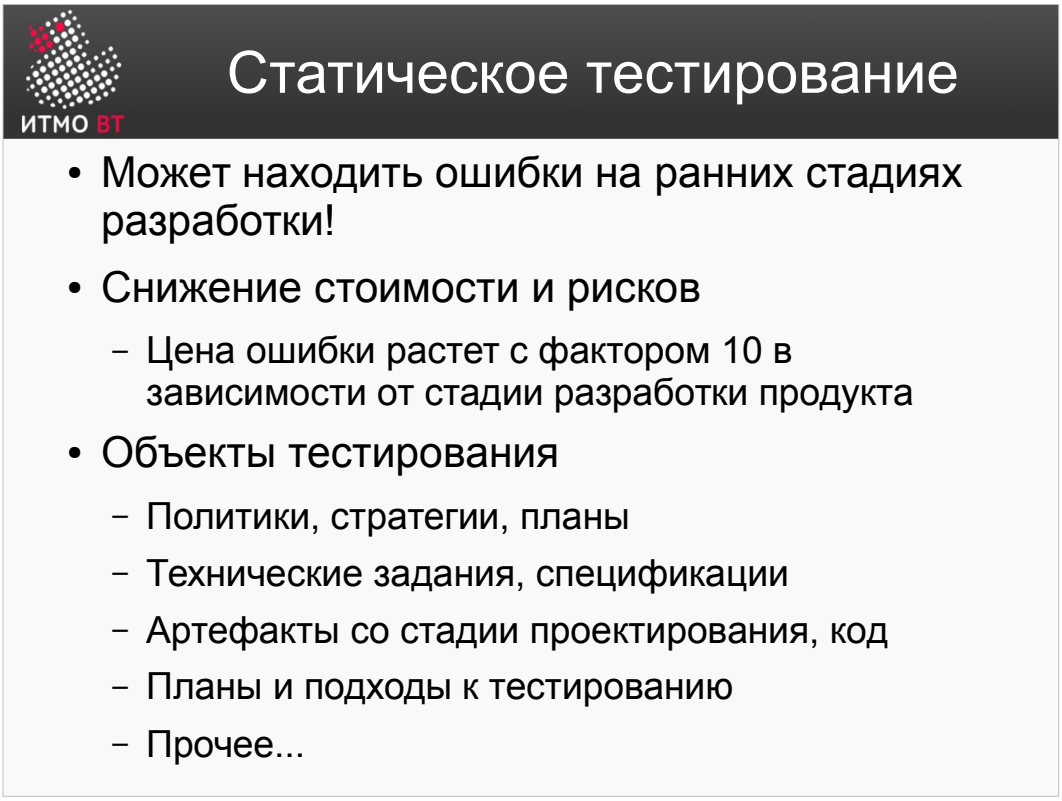

ИТМО ВТ
Автоматизация - не замена ручному тестированию
!!!


ИТМО ВТ
Статическое тестирование
4



Рецензирование - вид тестирования ПО (включая код), который может проводиться перед динамическим тестированием. Исправление дефектов, обнаруженных во время рецензирования на ранних этапах жизненного цикла ПО (например, дефектов, найденных в требованиях), часто обходится значительно дешевле по сравнению с дефектами, найденными во время выполнения тестов и исполнения кода.

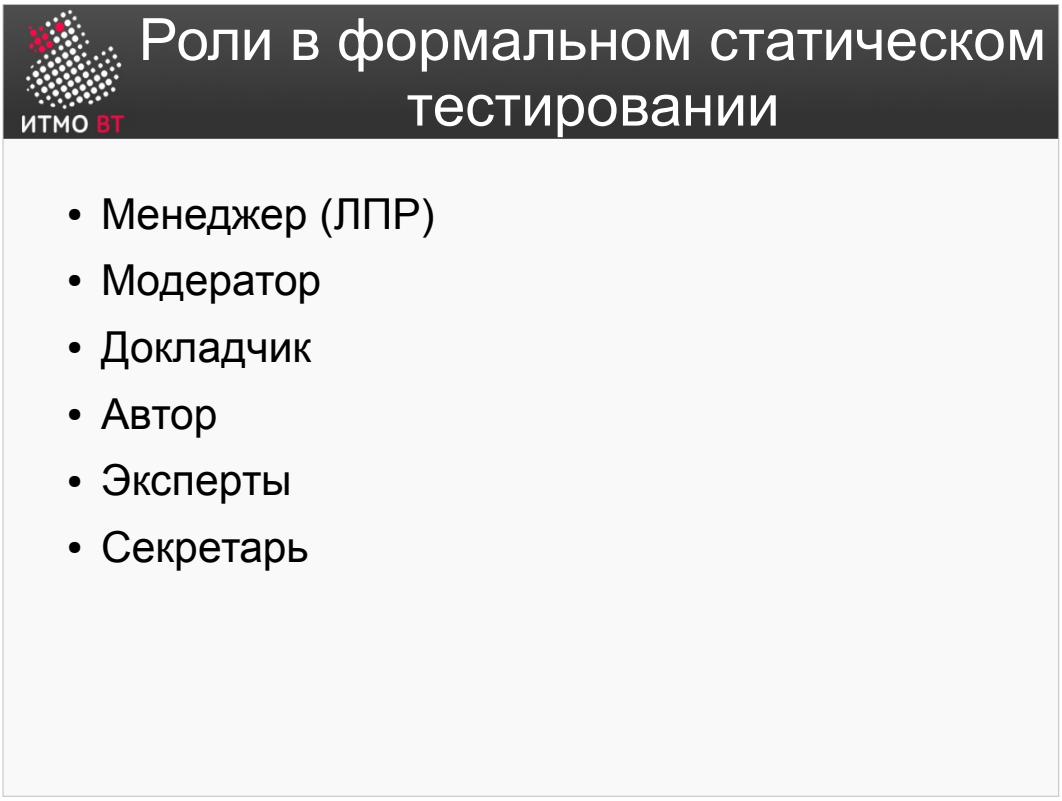
Рецензирование может проводиться вручную или с помощью специальных программных средств. Главной составляющей ручного процесса является исследование и комментирование программного продукта. Рецензирование может проводиться для любого продукта, связанного с разработкой ПО, включая спецификации требований и дизайна, код, планы тестирования, спецификации тестирования, тестовые сценарии, руководства пользователя, а также веб- страницы.



Преимущества рецензирования: раннее обнаружение и исправление дефектов, улучшение продуктивности разработки, уменьшение времени разработки, уменьшение времени и стоимости тестирования, сокращение стоимости жизненного цикла, уменьшение числа дефектов и улучшение коммуникаций. Во время рецензирования могут быть найдены упущения, например, в требованиях, которые маловероятно найти во время динамического тестирования.

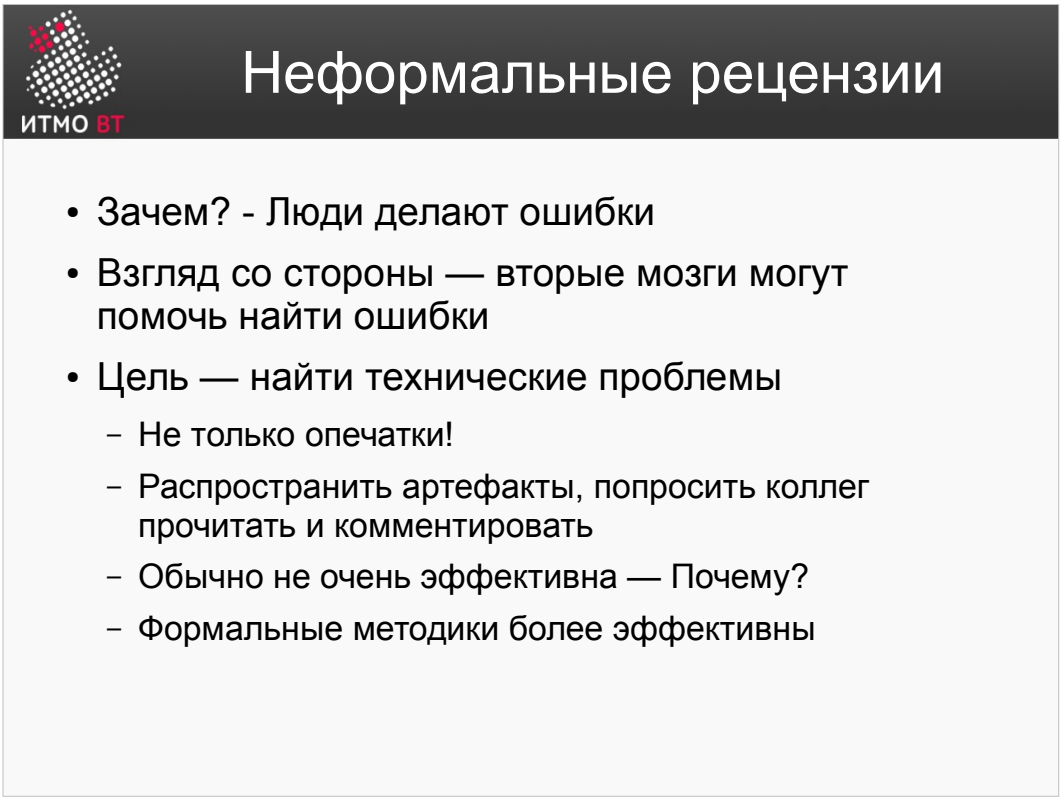
У рецензирования, статического анализа и динамического тестирования общая цель – обнаружение дефектов. Методы дополняют друг друга, так как с разной эффективностью находят различные типы дефектов. В отличие от динамического тестирования статические методы находят причины сбоя (дефекты), а не сами отказы.

Типичные дефекты, которые легче найти при рецензировании, чем при динамическом тестировании: отклонения от стандартов, дефекты в требованиях или дизайне, недостаточная пригодность к сопровождению и некорректные спецификации интерфейса.

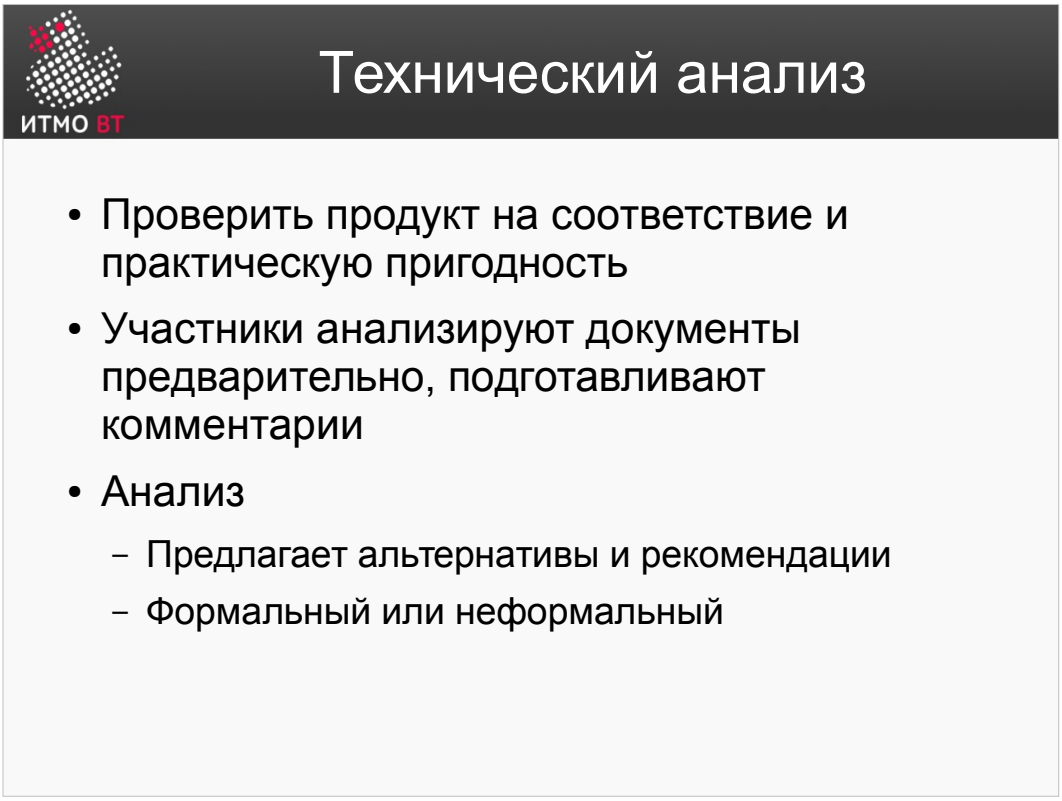


Обычно формальное рецензирование включает следующие роли:

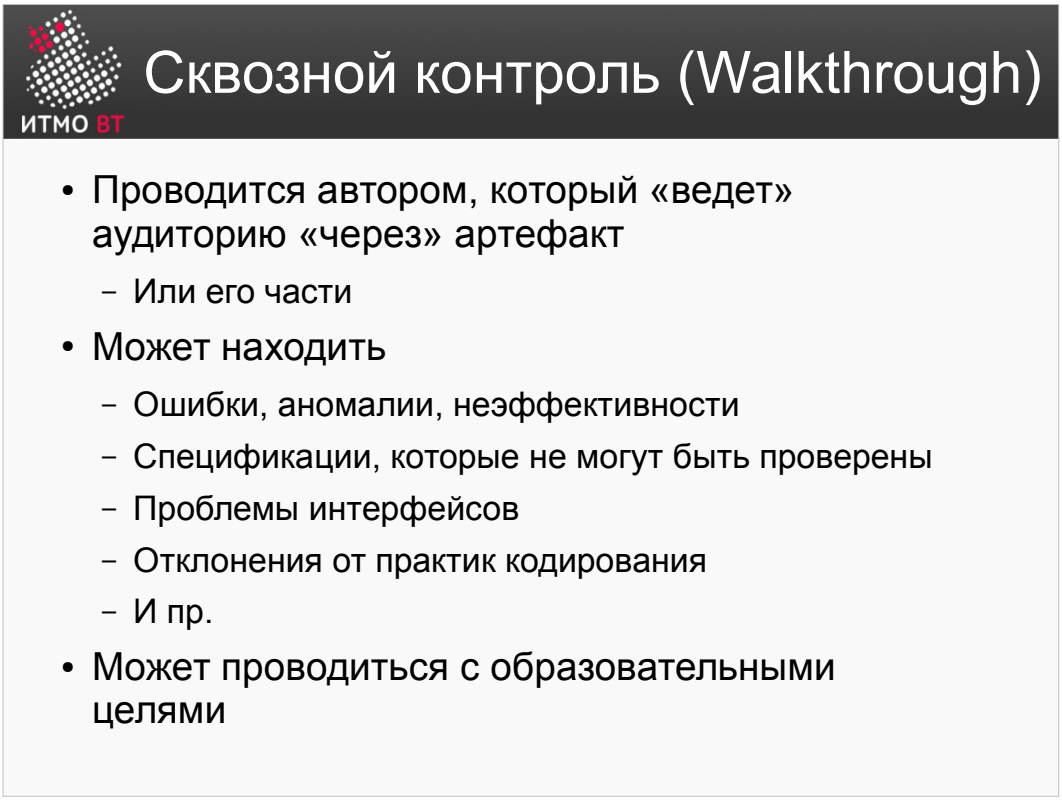
- Менеджер: принимает решение о проведении рецензирования, выделяет время в графике проекта и определяет, были ли достигнуты цели рецензирования
- Модератор: руководит проведением рецензирования документа или набора документов, включая планирование рецензирования, проведение встречи и отслеживание. При необходимости, модератор становится посредником между различными точками зрения и часто от него зависит успешное завершение рецензирования.
- Автор: автор или главный ответственный за документ(ы) для рецензирования
- Эксперты: люди со специальным техническим или бизнес опытом и знаниями (часто называются проверяющими или инспекторами), которые после необходимой подготовки, определяют и описывают проблемы и вопросы, найденные в проверяемом ПО. Экспертов выбирают таким образом, чтобы они представляли различные точки зрения и роли в процессе рецензирования. Эксперты должны принимать участие во всех встречах экспертов.
- Секретарь: документирует все проблемы и открытые вопросы, которые были определены во время экспертной встречи



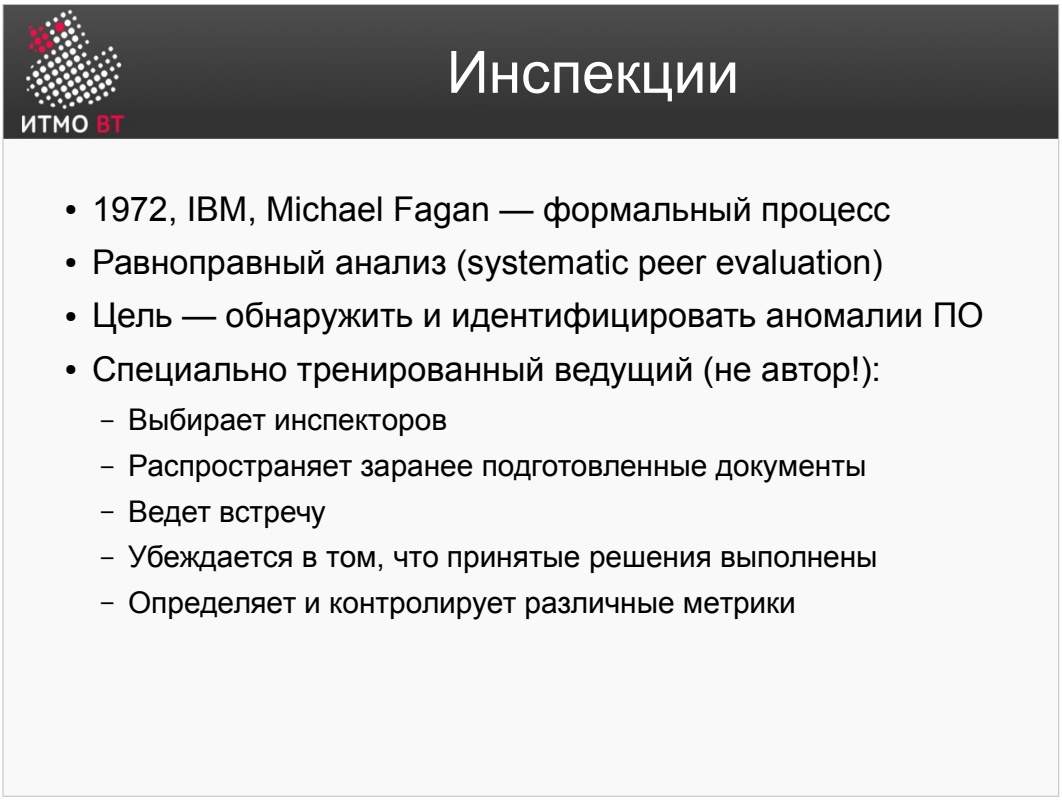
Неформальное рецензирование
• Отсутствие формального процесса
• Может принимать форму парного программирования или рецензирования
дизайна или кода техническим руководителем
• Результаты могут быть документированы
• Эффективность зависит от экспертов
• Главные цели: результат при минимуме затрат



Технический анализ
• Документированный процесс определения и нахождения дефектов, который включает участников команды, технических экспертов и, возможно, руководство
• Может проводиться как равноправный анализ без участия руководства
• В идеале проводится обученным модератором (не автором)
• Предварительная подготовка экспертов
• Подготовка отчета о рецензировании, который включает списонайденных проблем и вопросов, заключение, соответствует ли программный продукт требованиям, а также необходимые рекомендации
• На практике могут варьироваться от сильно до строго формальных
• Главные цели: обсуждение, принятие решений, оценка альтернатив, нахождение дефектов, решение технических проблем и проверка соответствия требованиям, планам, нормам и стандартам



Сквозной контроль
• Встреча проводится автором
• Может быть в форме сценариев, сухих прогонов, участия членов команды
• Не ограниченные по времени сессии
o Необязательно - подготовка экспертов перед встречей,
o Необязательно - подготовка отчета по рецензированию с записью
найденных проблем вопросов
• Наличие секретаря (не автора) необязательно
• На практике могут варьироваться от неформальных до строго формальных
• Главные цели: обучение, достижение понимания, поиск дефектов



Инспекция
• Проводится обученным модератором
• Обычно сопровождается равноправным исследованием
• Роли определены
• Включает сбор метрик
• Формальный процесс основан на правилах и контрольных списках
• Определены критерии входы и выхода для приемки программного продукта
• Предварительная подготовка перед встречей
• Инспекционный отчет включает список найденных проблем и вопросов
• Формальный процесс отслеживания
о Необязательно - процесс улучшения компонентов
• Наличие рецензента необязательно
• Главная цель: поиск дефектов

Сквозной контроль, технический анализ или инспекция могут проводиться внутри группы профессионалов, например, коллег одного организационного уровня. Такой тип рецензирования называется равноправным анализом.

Инспекции - 2

- Четко определенные шаги
  - Вход
  - Планирование
  - Обзор
  - Подготовка
  - Обсуждение
  - Переработка
  - Выработка рекомендаций (follow up)
  - Выход

Могут повторяться

- Повторяется до тех пор, пока все участники включая модератора не удовлетворены

Инспекции — входные и выходные критерии

- Должны быть обязательно определены
  - Не тратить время попусту неподготовленной встречей
- Ведущий убеждается в этом перед началом
- Входные критерии могут быть такими:
  - Подготовлены чеклисты
  - Основополагающие документы вышли из инспекций с известным уровнем ошибок
  - За 10 минут пробной проверки был найден не более одного найденного дефекта
  - Документы грамматически проверены и пр...
- Выходные критерии — документы согласованы

Планирование инспекции

- Зависит от инспектируемых артефактов
  - Учитывать размер, сложность. Определяется «Check-rate»
  - Разбиение на одновременно обрабатываемые «куски»
- Зависит от участников
  - Убедиться, что все будут присутствовать
  - Обладают ли определенными знаниями и опытом
- Собираемые метрики (н-р: размеры, кол-во найденных дефектов, цена изменений, время на проверку)
- Разрабатывается расписание

Инспекция: обзор

- Включает:
  - Групповое обучение инспекторов
  - Назначение ролей инспекторам
  - Распространение материалов
- Ведущий представляет инспекторов

ИТМО ВТ

Инспекция: подготовка

- Сердце инспекции — здесь проводится само тестирование
- Инспекторы проверяют артефакты в соответствии с ролью
  - Отмечают время начала
  - Отмечают и записывают проблемы, фокусируясь на основных
  - Используют точно отведенное время (не больше, не меньше — check rate)
  - Классифицируют и считают проблемы (Major, minor, ?)

Инспекторы - роли

- Человек может фокусироваться только на двух проблемах одновременно
- Роли могут быть выбраны из следующих:
  - Чеклист — инспектор проверят по нему
  - Документы — инспектор проверяет целостность между несколькими документами
  - Фокус — поиск выделенных проблем
  - Перспектива — представить роль и будущее пользователя
  - Процедура — инспектор следует особой процедуре (.)
  - Сценарий — следование заданному сценарию (более специфично, чем предыдущий)
  - Стандарт — проверка на соответствие стандартам
  - Точка зрения — инспектирует с т.з. н-р пользователя

Инспекция - встреча, обсуждение

- Основное назначение — протоколирование результатов подготовки
  - Позволяет найти еще 10-20% проблем
  - Проблема: id, (Mmon), описание, документ, страница
- Ведущий:
  - Дипломатичен, краток, категоричен
  - Не позволяет развивать дискуссию, длительно предлагать решения
  - Принимает непопулярные решения (удалить к-л, остановить встречу)
- Ограничена по времени — не более 2-х часов
- Решение по артефакту — «принят», «на переработку», «отклонен»

Инспекция — переработка и выработка рекомендаций

- Если артефакт не принят - отправляется на переработку
  - Автором
  - Редактором
  - Другими работниками
- Инспекция не может завершиться, пока не удовлетворены выходные критерии
  - Проверяется ведущим
- Готовый артефакт передается в систему контроля версий для остальной команды

Статические методы. Обзор

	Сквозной контроль	**Технический Анализ**	**Инспекция**
Основное Предназначение	Поиск дефектов	Поиск дефектов	Поиск дефектов
Дополнительная цель	Обмен опытом	Принятие решений	Улучшение процесса
Подготовка	Обычно нет	Популяризация	Формальная подготовка
Ведущий	Автор	В зависимости от обстоятельств	Подготовленный модератор
Рекомендованный размер группы	2-7	>3	3-6
Формальная процедура	Обычно нет	Иногда	Всегда
Объем материалов	небольшой	От среднего до большого	небольшой
Сбор метрик	Обычно нет	Иногда	Всегда
Выходные данные	Неформалный отчет	Формальный отчет	Список дефектов, результаты метрик, формальный отчет

Средства статического анализа кода

- Большое количество (C - Lint, Java — FindBugs)
- Находят:
  - Неопределенное поведение (переменная не инициализирована)
  - Нарушение алгоритмов использования библиотеки (fopen без fclose)
  - Сценарии некорректного поведения
  - Переполнение буфера
  - Разрушение кроссплатформенности
  - Дефекты копи-пейста
  - …


ИТМО ВТ
Системное тестирование
5

Тестирование системы в целом

- Начинается после окончания интеграции
- Включает несколько фаз:
  - Системное тестирование — выполняется внутри организации-разработчика
  - Альфа- и Бета-тестирование — выполняется пользователем под контролем разработчика
  - Приемочное тестирование — выполняется пользователем.
    - Результат — платить или не платить.
- Методики практически одинаковые, различная строгость интерпретации результатов.

Системное тестирование

- Стандартная цель
- Фокус:
  - Функциональность системы с точки зрения пользователя
  - Нефункциональные характеристики
- Проводится валидация!
  - В основном метод черного ящика

Системное тестирование. Напоминание!

- Тестирование — это как научный эксперимент — необходимо повторение
- Автоматизация, скриптинг
- Результаты должны быть:
  - Записаны в журнал.
  - На каждый найденный дефект — зарегистрирован инцидент
  - Периодические отчеты для найденных ошибок

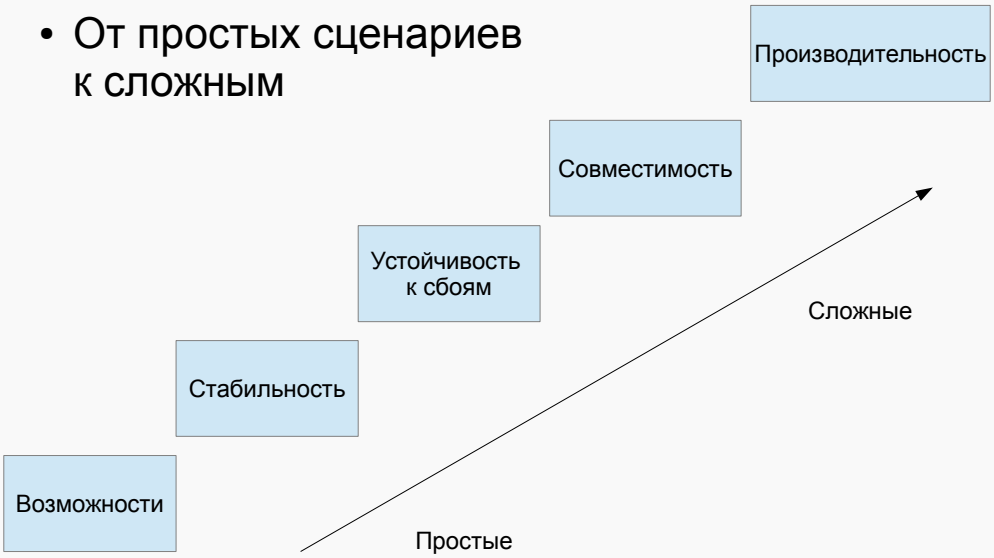
Контекст системного тестирования

- Тестовое окружение д.б. максимально близко к операционному
  - Оборудование
  - Операционная система
  - Сетевое окружение
  - В аппаратуре — частота, тайминги … и пр.
- Тестирование эмулирует реальные действия пользователей
  - Тестеры исполняют роли реальных пользователей
  - Скрипты исполняют реальные сценарии
- Система под управлением разработчиков

ИТМО ВТ

Где начать?

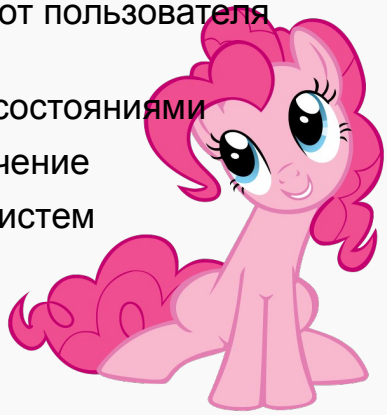
- От простых сценариев к сложным



ИТМО ВТ

Возможности

- Работают ли основные функции системы
  - Один пользователь
  - Одна транзакция
  - Корректный ввод информации от пользователя или из файлов, БД
  - Нормальные переходы между состояниями
  - «Среднее» аппаратное обеспечение
  - Положительные проверки подсистем безопасности



Стабильность

- Попробуем более реалистичную ситуацию
  - Все еще валидные входные данные и транзакции
  - Несколько пользователей одновременно
  - Реальные последовательности транзакций
  - Одновременно несколько транзакций от одного пользователя (если есть)
  - Запуск системы на длительный период
    - Утечки памяти, переполнения диска. ….

Устойчивость к сбоям

- Let's break it!
  - Пользователи вводят неправильные данные
  - Сбой сети, испорченные файлы
  - Неправильные переходы между состояниями
  - Выполнение операций в неправильном порядке
  - Аварийное отключение систем/подсистем
  - Намеренные ошибки подсистем безопасности
    - Плохие пароли, внешний взлом

ИТМО ВТ

Совместимость

- Проверяется функционирование в разных средах
  - Взаимодействие с локальными и удаленными системами
  - Проверка функционирования с различными версями
    - Библиотек
    - Браузеров
    - Операционных систем

Производительность - CARAT

- CARAT
  - Capacity — Нефункциональные возможности
  - Accuracy — Точность
  - Responce Time — Время ответа
  - Availability — Готовность
  - Throughput — Пропускная способность

Нефункциональные возможности

- Максимальное количество (пользователей, записей в БД, файлов, Кб, ГГц), поддерживаемое системой
- Одновременно, не нарушая других требований производительности

Точность

- Корректность:
  - Алгоритмов
  - Результатов
- Может быть критичной для определенного класса систем

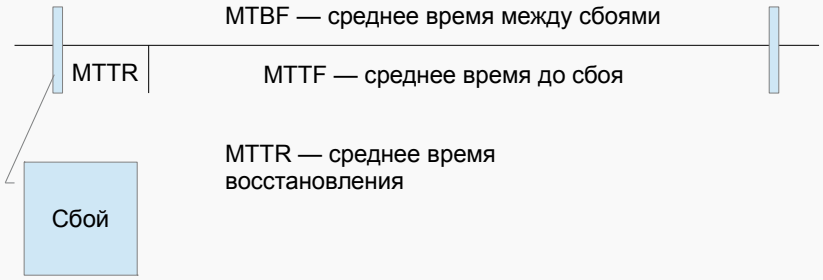
Время отклика

- Один из наиболее часто измеряемых и критичных критериев
  - Системы реального времени
- От «нажатия на кнопку» до «полной загрузки страницы»
- От подачи управляющего сигнала до наступления реакции
- Под минимальной, расчетной, пиковой нагрузкой

ИТМО ВТ

Готовность

- Коэф. готовности = (MTBF — MTTR)/ MTBF



- 0,99999 — сколько минут в год?

Пропускная способность

- Количество операций (транзакций) в секунду которые может поддерживать система
- Баланс с временем отклика
- Стресс-тестирование — последовательная нагрузка системы до момента неприемлемого времени отклика

Инструменты нагрузочного тестирования

- HP Load Runner
- LoadComplete
- IBM Rational Performance Tester
- Load UI Pro
- Apache Jmeter
- The Grinder
- Tsung

Нагрузочное тестирование с Jmeter

- Бесплатный, кроссплатформенный
- Интерфейс пользователя
  - Простой, строит графики
- Возможность создания распределенной нагрузки
- Эмуляция одновременной работы пользователей
- Снятие метрик
- Возможность разрабатывать плагины
- Планы тестирования в XML — просто контролировать версии

http://jmeter.apache.org

Jmeter и функциональное тестирование

- Может быть использован для записи запросов
- Jmeter → Функциональное тестирование → Нагрузочное тестирование
- Jmeter — не браузер!

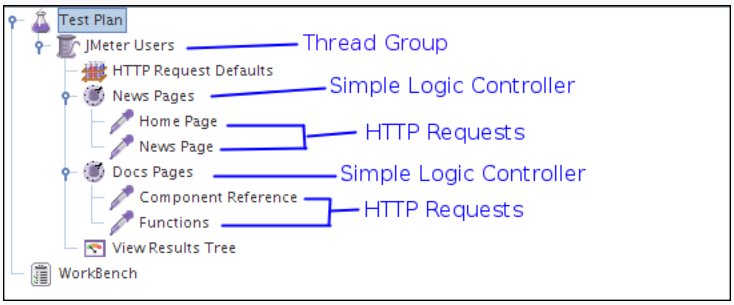
Тестовый план

- Thread Group
  - Описывает пул пользователей для выполнения теста
  - Количество, возрастание и пр..
- Семплеры
  - Формируют запросы, генерируют результаты
  - Большой набор встроенных протоколов (TCP, HTTP, FTP, JDBC, SOAP, JMS, SMTP, …..)

http://jakarta.apache.org/jmeter/usermanual/component_reference.html

Тестовый план - 2

- Логические контроллеры
  - Определяют порядок вызова семплеров
  - Конструкции управления (if, loop, …)
  - Управление потоком



http://jakarta.apache.org/jmeter/usermanual/component_reference.html

ИТМО ВТ

Тестовый план - 3

- Слушатели
  - Получают ответы
  - Осуществляют доп. операции с результатами: просмотр, запись, чтение и др.
  - Не обрабатывают данные! (в командной строке, нужен GUI)
- Таймеры
  - Задержки между запросами
  - Постоянные, в соответствии с законами

http://jakarta.apache.org/jmeter/usermanual/component_reference.html

Тестовый план - 4

- Assertion
  - Проверяют результаты
- Элементы конфигурации
  - Сохраняют предустановленные значения для семплеров
- Препроцессоры
  - Изменяют семплеры в их контексте (HTML Link Parser)
- Постпроцессоры
  - Применяются ко всем семплерам в одном контексте

http://jakarta.apache.org/jmeter/usermanual/component_reference.html

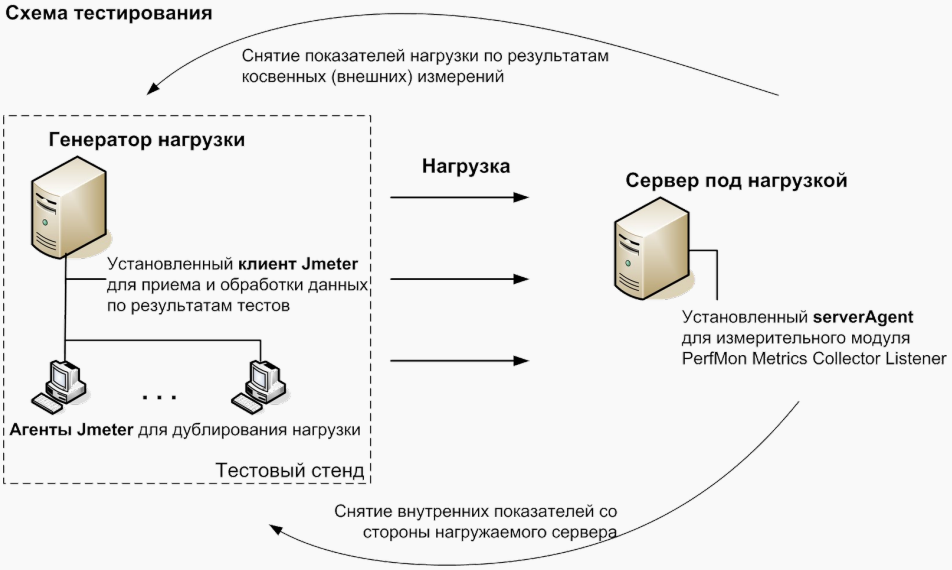
Тестовый план — порядок выполнения

- Конфигурационные элементы
- Препроцессоры (Pre-Processors)
- Таймеры (Timers)
- Семплеры (Sampler)
- Постпроцессоры (Post-Processors)
- Assertions
- Слушатели (Listeners)

Jmeter — дополнительные возможности

- Автоматическая генерация CSV-файла
- Bandwidth Throttling - ограничение полосы пропускания
  - Статическое (CPS), динамическое (error rate)
- IP Spoofing — замена src_ip, для разделения клиентов
  - Проверка балансировщиков нагрузок
- Поддержка теста TPC-C
  - Стандартный тест на OLTP нагрузку

Jmeter — распределенное тестирование



http://forworktests.blogspot.ru/2013/03/jmeter.html

Альфа-тестирование

- Проводится небольшим количеством пользователей внутри организации, разработавшей ПО
  - Может проводиться до окончания системного тестирования
  - Ранние отзывы пользователей об использовании системы в рабочем окружении

Бета-тестирование

- Проводится выбранной группой пользователей в своем/рабочем окружении
- Windows Beta test — по всему миру
- Могут быть ошибки
- Может быть не реализован весь функционал
- Ранние отзывы для разработчиков
- Превью для пользователей

Альфа- и бета-тестирование

- Проводится неформально
  - Пользователи работают с тем функционалом, который им интересен
  - … отмечают то, что не работает
  - … доводят дефекты и сбои до разработчика
- Нет тестовых сценариев и планов
- Разработчики не гарантируют, что немедленно исправят проблемы. Исправления могут быть:
  - в финальном релизе;
  - в следующей бета-версии.

Приемочное тестирование

- Ключевой аспект — управляется пользователями
  - А не разработчиками
- Проверка системы на «соответствие предназначению и практическому использованию»
- Формальное
- Неформальное

Приемочное тестирование - 2

- Все ли требования удовлетворены
  - Функциональные
  - Нефункциональные
- Источник — документ «требования к системе» (RUP — SRS)
  - Подготовка может начинаться на ранних этапах
  - Может применяться статическое тестирование
- Ответственно подходить к выбору пользователей и обязательно их обучить!

Приемочное тестирование - 3

- Окружение должно быть полностью готово
  - Аппаратура и программное окружение
  - Может быть первый запуск в рабочем окружении
- Тесты проводятся и анализируются так же, как и в системном тестировании