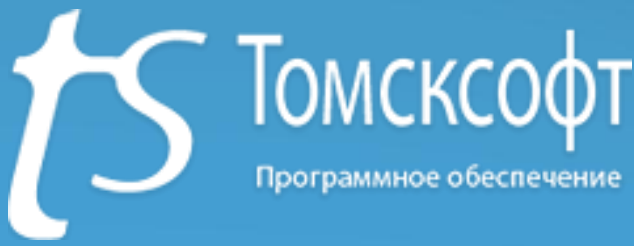
Автоматическое тестирование мобильных приложений на основе скриншотов

Сергей Борисов, tech lead, ТомскСофт
Иван Лебедев, интерн, ТомскСофт



Об авторах

Сергей Борисов

В настоящее время:
- Руководитель проекта в ТомскСофт
- ст. преп-ль в ТУСУР, каф. КСУП

Ранее:
- MobilTeck
- SiberLogic
- ЭлеСи
- ТМЦДО ТУСУР
- НИИ АЭМ

Общий стаж в разработке ПО: 19 лет
Общий стаж в высшем образовании: 15 лет

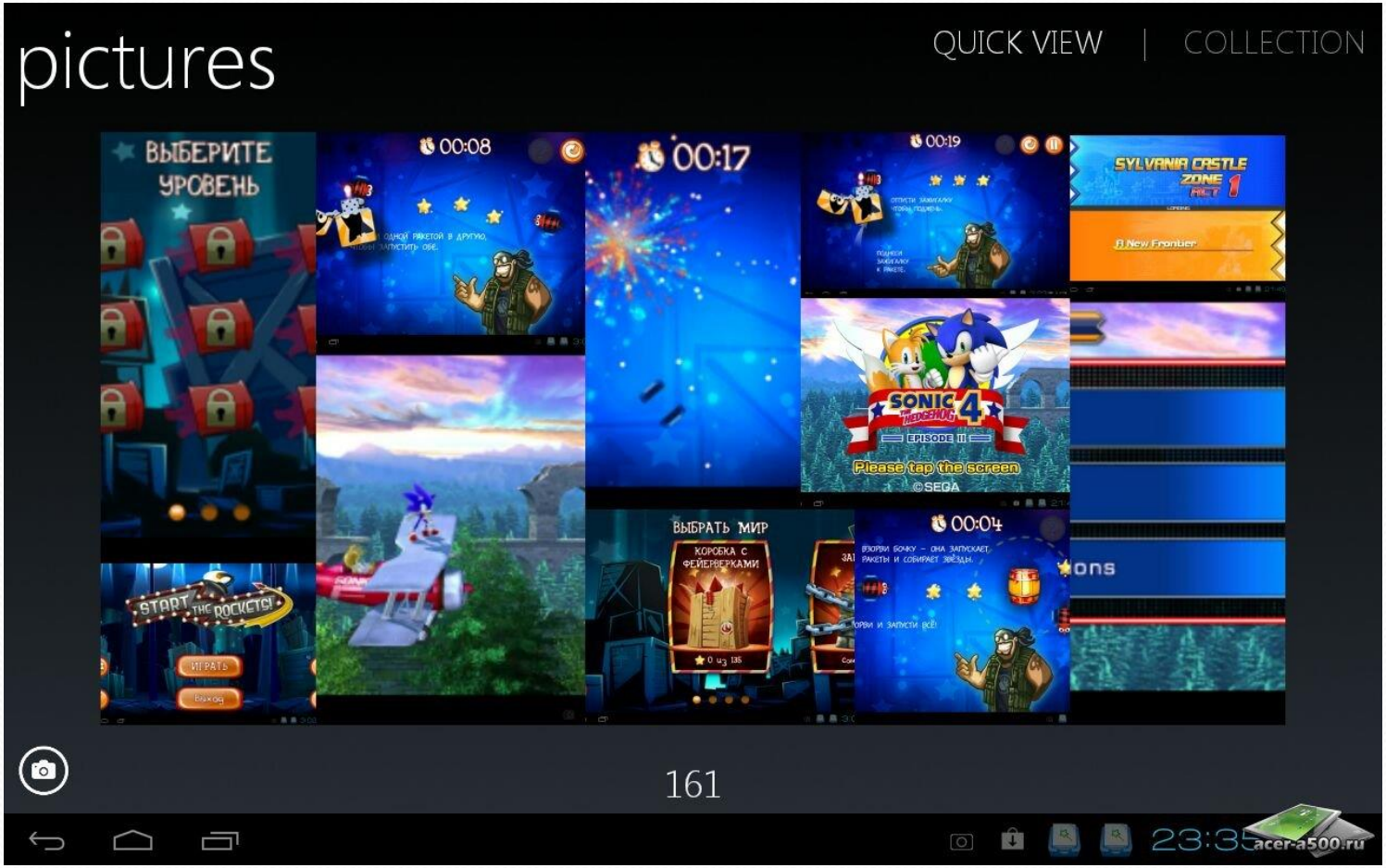
Об авторах

Иван Лебедев

В настоящее время:
- Интерн в ТомскСофт
- студент 5-го курса ТУСУР, каф. КСУП

Суть проблемы

1. Приложения обладают UI (особенно мобильные)



Суть проблемы

2. UI включает множество различных элементов с разным форматированием

Суть проблемы

3. Зоопарк различных разрешающих способностей

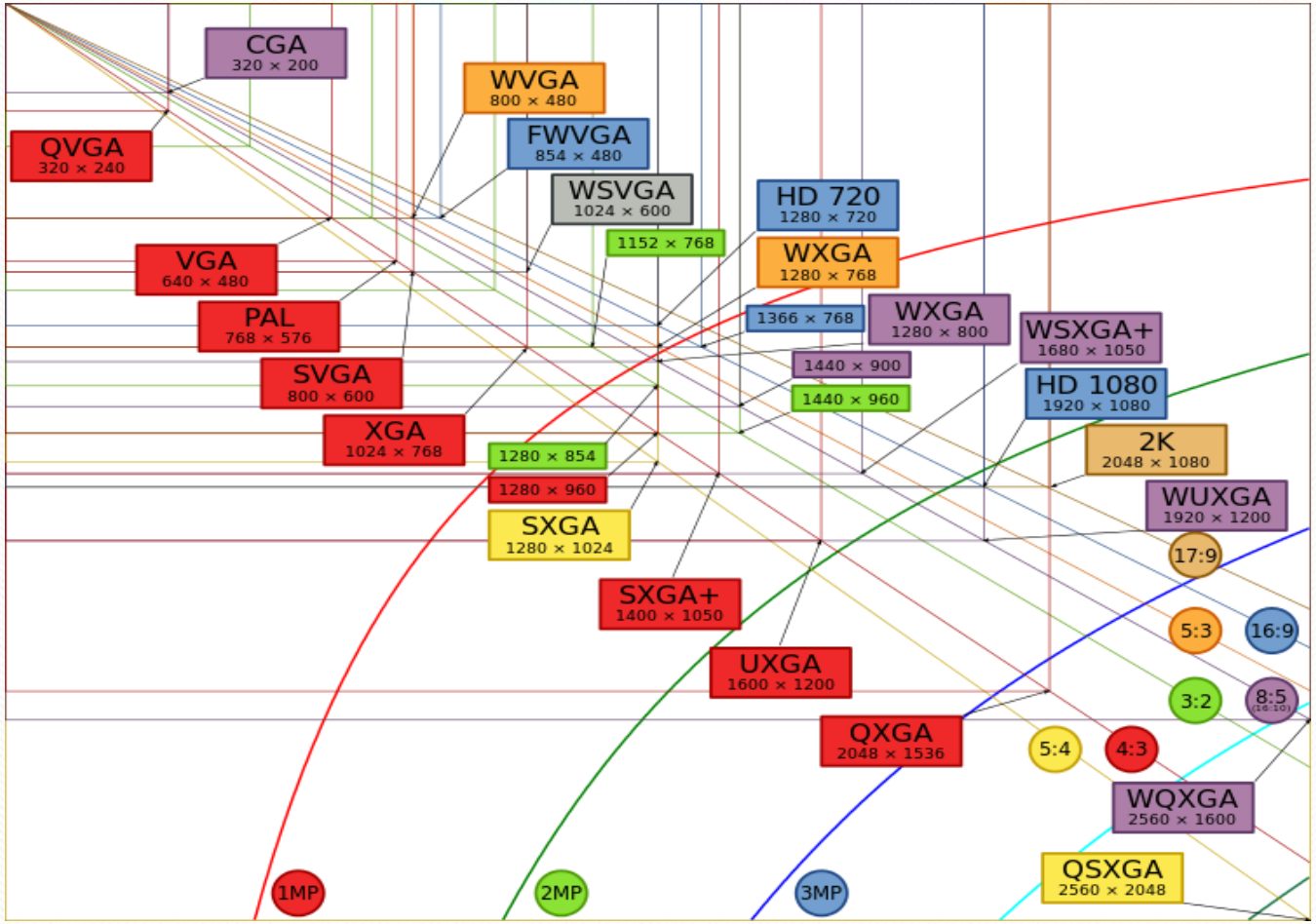
Суть проблемы

3. Зоопарк различных разрешающих способностей

	120 точек на дюйм	160 точек на дюйм	240 точек на дюйм	320 точек на дюйм
Маленький	240x320		480x640	
Нормальный	240x400 240x432	320x480	480x800 480x854 600x1024	640x960
Большой	480x800 480x854	480x800 480x854 600x1024		
Очень большой	1024x600	1280x800 1024x768 1280x768	1536x1152 1920x1152 1920x1200	2048x1536 2560x1536 2560x1600

Суть проблемы

4. Зоопарк соотношений экрана



Тестирование

- Если вы разрабатываете приложение по принципу: “сделали – отдали – забыли”, то вам не нужно тестирование

- Если вы разрабатываете приложение, которое собираетесь развивать и поддерживать, то вам нужно тестирование

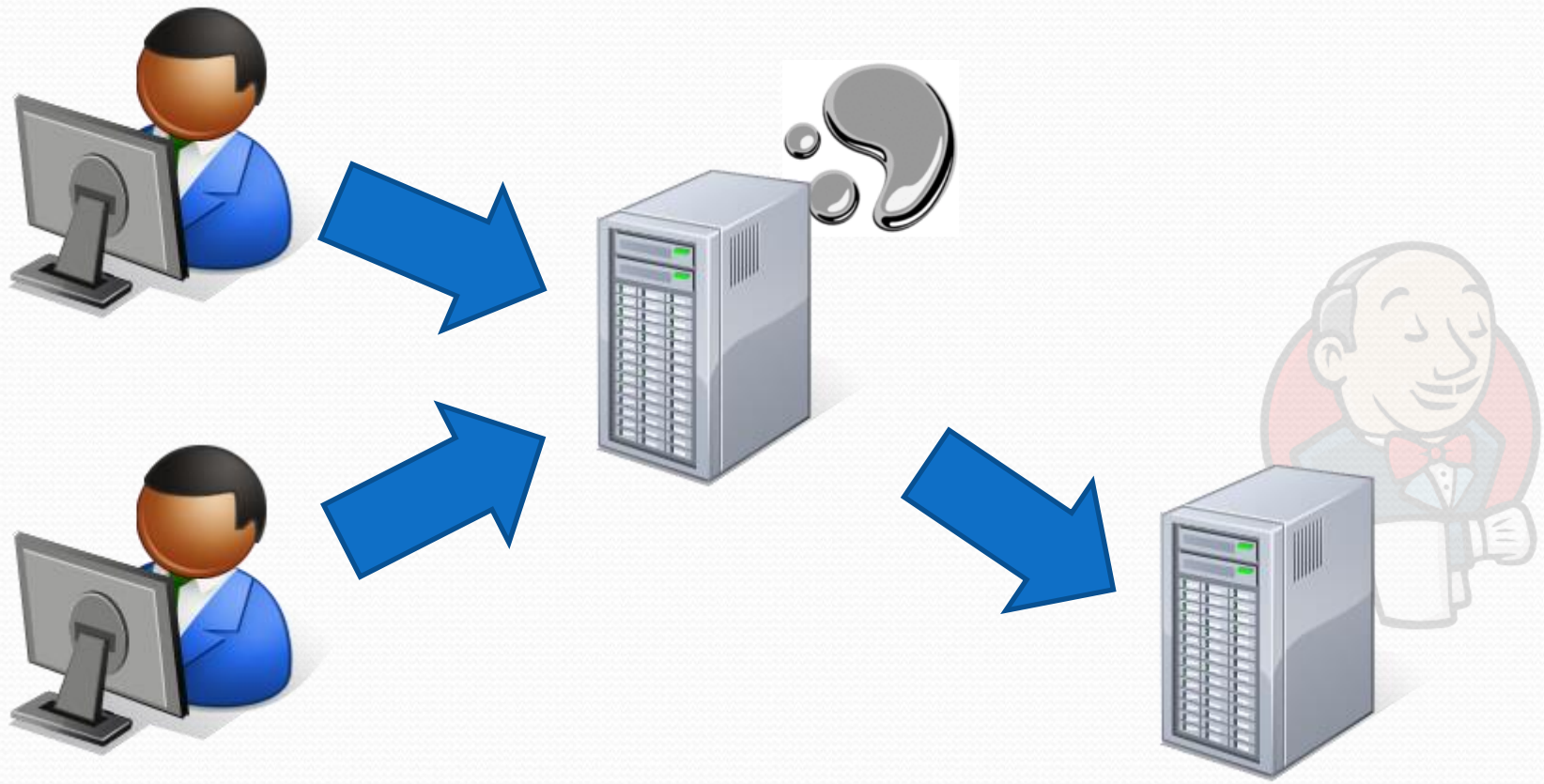
Классификация тестирования: объект тестирования

- Функциональное тестирование (functional testing)
- Тестирование производительности (performance testing)
- Юзабилити-тестирование (usability testing)
- Тестирование интерфейса пользователя (UI testing)
- Тестирование безопасности (security testing)
- Тестирование локализации (localization testing)
- Тестирование совместимости (compatibility testing)

Классификация тестирования: по времени проведения

- Альфа-тестирование (alpha testing)
  - Тестирование при приёмке (smoke testing)
  - Тестирование новой функциональности (new feature testing)
  - Регрессионное тестирование (regression testing)
  - Тестирование при сдаче (acceptance testing)
- Бета-тестирование (beta testing)

Автоматизированное тестирование

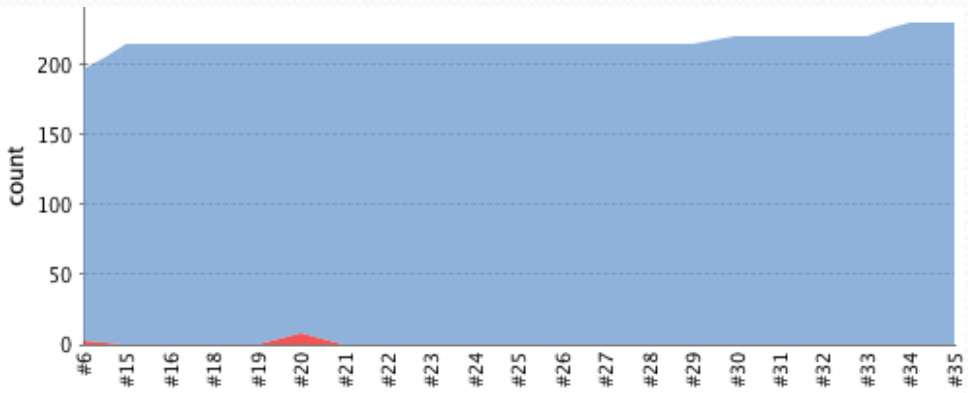


Разработчики Репозитарий CI сервер

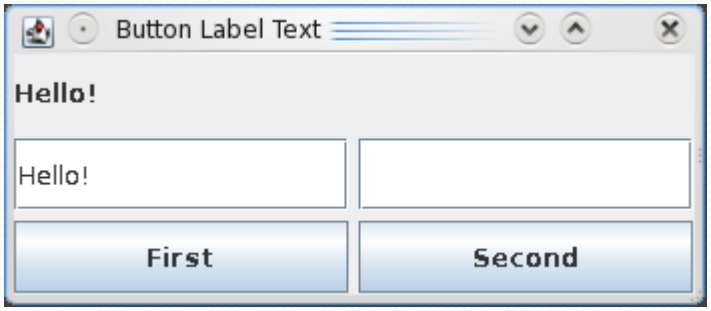
Регрессионное автоматическое тестирование

#35	Mar 13, 2013 12:16:44 PM
#34	Mar 12, 2013 9:51:41 PM
#33	Mar 12, 2013 5:31:45 PM
#32	Mar 7, 2013 3:31:48 PM
#31	Mar 6, 2013 2:46:47 PM
#30	Mar 4, 2013 6:01:47 PM
#29	Mar 4, 2013 5:46:47 PM
#28	Mar 1, 2013 5:46:48 PM
#27	Feb 28, 2013 9:01:47 PM
#26	Feb 28, 2013 12:16:47 PM
#25	Feb 27, 2013 1:16:47 PM
#24	Feb 26, 2013 12:01:47 PM
#23	Feb 26, 2013 11:46:47 AM
#22	Feb 26, 2013 11:31:47 AM
#21	Feb 22, 2013 5:31:02 PM
#20	Feb 22, 2013 5:22:27 PM
#19	Feb 22, 2013 5:02:26 PM
#18	Feb 22, 2013 4:47:20 PM
#17	Feb 22, 2013 4:46:52 PM
#16	Feb 22, 2013 1:16:02 PM
#15	Feb 22, 2013 10:16:03 AM

- Модульные тесты
- Функциональные интеграционные тесты
- Тесты UI

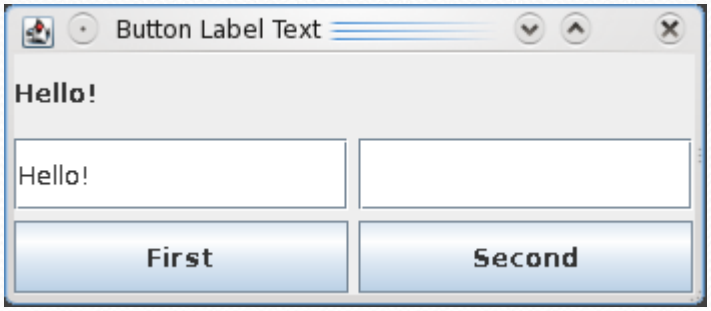


Тестирование UI



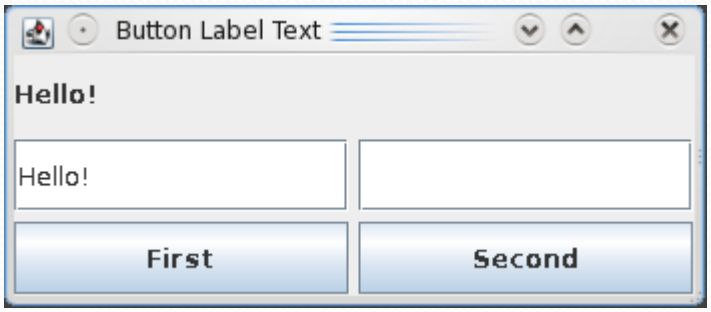

```
import org.netbeans.jemmy.*;
import org.netbeans.jemmy.operators.*;

import java.awt.*;

public class MainTest {

    public int runIt(Object param) {
        try {
            //Запуск Swing приложения
            new ClassReference("swing.SwingApplication").startApplication();
            //Поиск JFrame
            JFrameOperator mainFrame = new JFrameOperator("SwingApplication");
            //Делаем небольшую задержку
            new QueueTool().waitEmpty(200);
            //Ищем кнопку по имени
            JButtonOperator firstButton = new JButtonOperator(mainFrame, "First");
            firstButton.push();
            //Ищем JTextField по имени после нажатия кнопки должно стать "First pressed"
            JTextFieldOperator textField = new JTextFieldOperator(mainFrame,"First pressed");
            //Очищаем поле ввода
            textField.clearText();
            textField.enterText("Hello");
            //После ввода текста на JFrame должен появиться JLabel
            //Метка должна быть "Hello"
            new JLabelOperator(mainFrame,"Hello");
            //Поиск второго компонента JTextField по индексу
            JTextFieldOperator textField2 = new JTextFieldOperator(mainFrame,1);
            textField2.enterText("Field2");
            //Опять ищем JLabel по тексту
            new JLabelOperator(mainFrame,"Field2");
            JButtonOperator secondButton = new JButtonOperator(mainFrame, "Second");
            assert(new Rectangle(10,10,50,20).equals(secondButton.getBounds()));
            assert("Second".equals(secondButton.getText()));
            assert(secondButton.isDefaultButton());
            secondButton.push();
            mainFrame.close();
        }
        catch(Exception e) {
            e.printStackTrace();
            return(1);
        }
        return(0);
    }
}
```

Тестирование UI




```
new ClassReference("swing.SwingApplication").startApplication();
//Поиск JFrame
JFrameOperator mainFrame = new JFrameOperator("SwingApplication");
//Делаем небольшую задержку
new QueueTool().waitEmpty(200);
//Ищем кнопку по имени
JButtonOperator firstButton = new JButtonOperator(mainFrame, "First");
firstButton.push();
//Ищем JTextField по имени после нажатия кнопки должно стать "First pressed"
JTextFieldOperator textField = new JTextFieldOperator(mainFrame,"First pressed");
//Очищаем поле ввода
textField.clearText();
textField.enterText("Hello");
//После ввода текста на JFrame должен появиться JLabel
//Метка должна быть "Hello"
new JLabelOperator(mainFrame,"Hello");
```

Тестирование UI



Тестирование UI по элементам – дело занудное

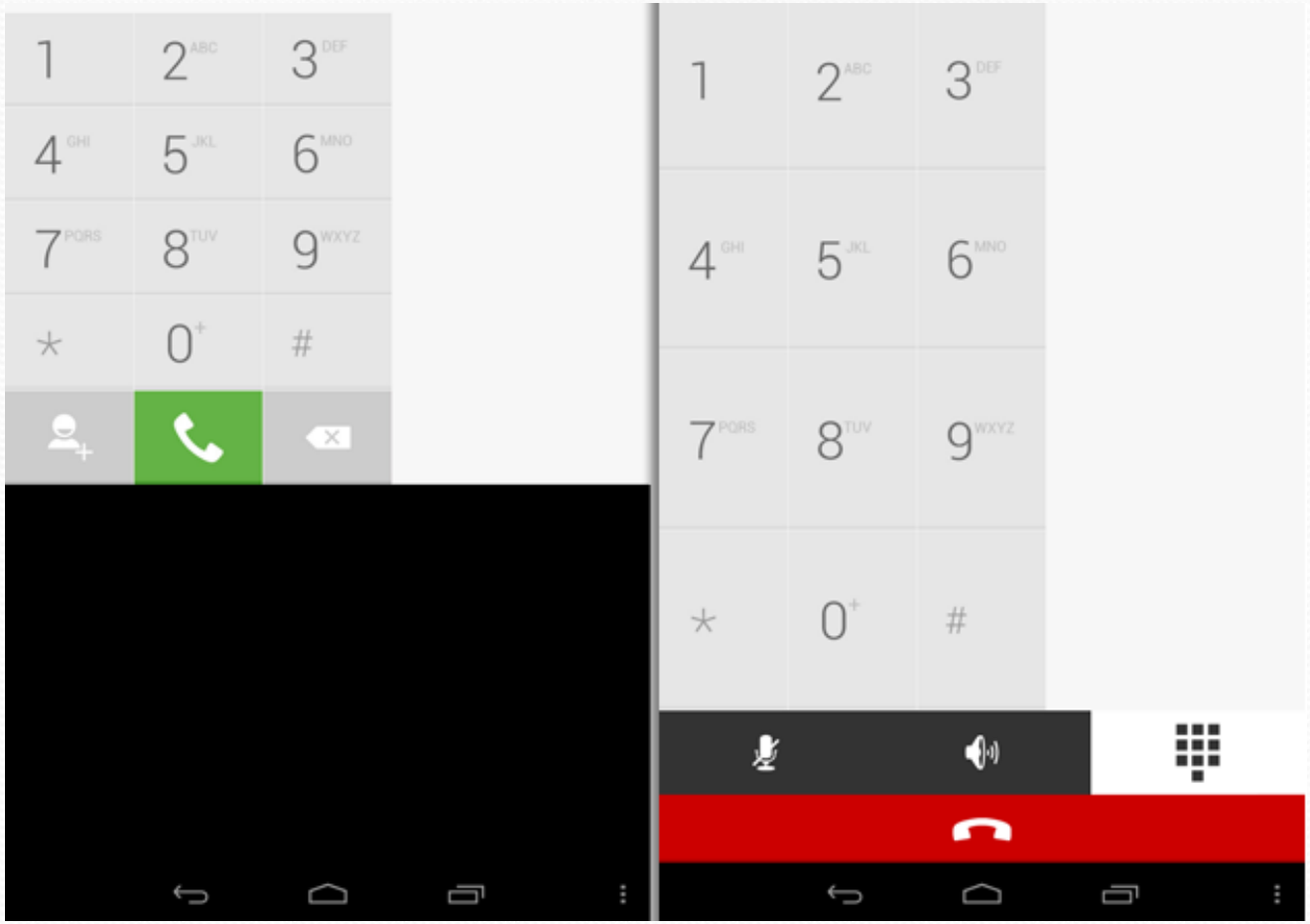
```
//Поиск второго компонента JTextField по индексу
JTextFieldOperator textField2 = new JTextFieldOperator(mainFrame,1);
textField2.enterText("Field2");
//Опять ищем JLabel по тексту
new JLabelOperator(mainFrame,"Field2");
JButtonOperator secondButton = new JButtonOperator(mainFrame, "Second");
assert(new Rectangle(10,10,50,20).equals(secondButton.getBounds()));
assert("Second".equals(secondButton.getText()));
assert(secondButton.isDefaultButton());
secondButton.push();
```

Вопросы на которые может ответить фунциональное тестирование через UI

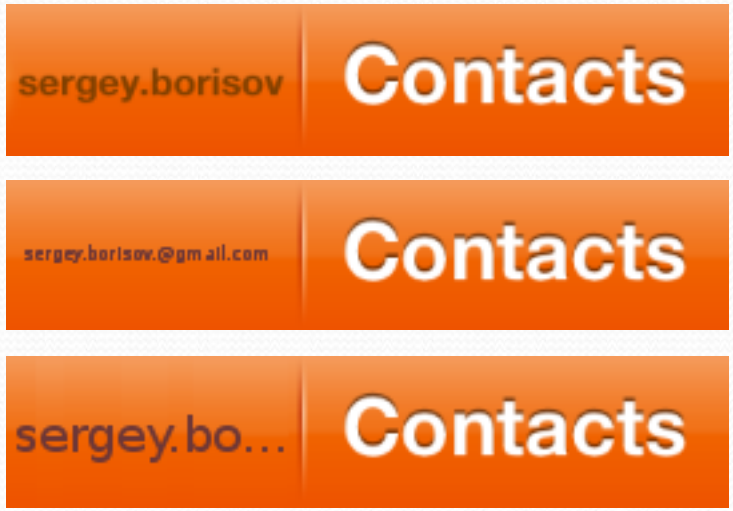
- Наличие/отсутствие элемента управления
- Видимость этого элемента управления
- Различные другие свойства объекта
- Корректность реакции элемента управления на действия пользователя

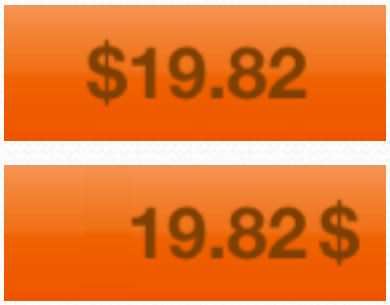
Вопросы на которые фунциональное тестирование через UI ответить не может

Находится ли элемент управления там, где того ожидал дизайнер?



Выглядит ли элемент управления, так как того ожидал дизайнер?

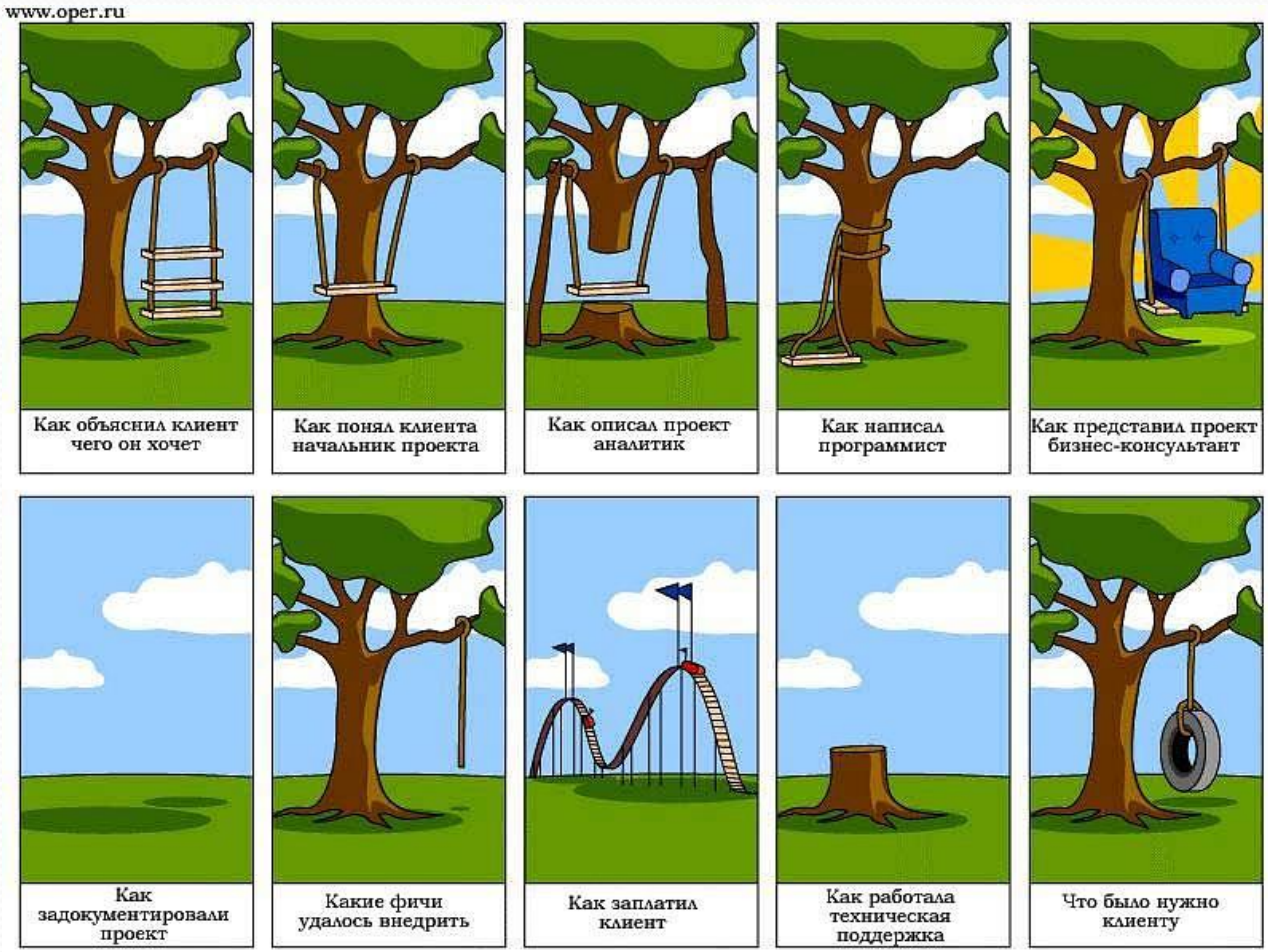




Или с каким конкретным набором данных он выглядит:

- слишком мелко
- не полностью
- и т.д.

Соответствует ли отображение интерфейса ожиданиям дизайнера?



А соответствует ли оно ожиданиям заказчика?

На эти вопросы может легко ответить человек

Надо всего лишь просмотреть все экраны…

Со всеми возможными данными…

На всех возможных разрешениях устройств…

На всех версиях операционной системы…

Объем работы тестировщика

Расчетное время

Количество экранов для разных наборов данных – 100

Количество разрешений экрана – 5

Время подготовки и изучения одного экрана – 5 минут

Итоговое время:

100x5x5=2500мин ~1 неделя

Для каждой версии ОС

Задача

Нужен фреймоворк, который позволяет:

- Делать автоматическое регрессионное тестирование приложения

- Учитывать все особенности отображения элементов управления

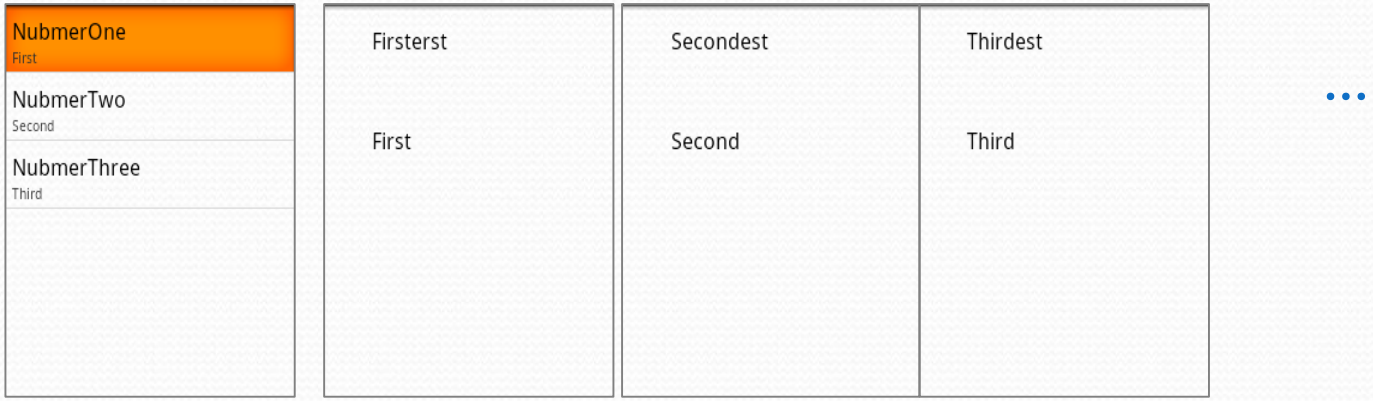
- Делать тесты на разных разрешениях экранов и разных версиях операционной системы

Решение: шаг 1

Для каждого экрана приложения нужно подготовить наборы тестовых данных, которые будут в нем отображаться

Screen2

Screen1 DataSet1 DataSet2 DataSet3



Решение: шаг 2

Создать тестовое приложение, которое в автоматическом режиме:

- Загружает заранее подготовленные данные
- Отображает экран
- Делает скриншот этого экрана

Примечание: функциональность элементов управления, в данном случае не важна

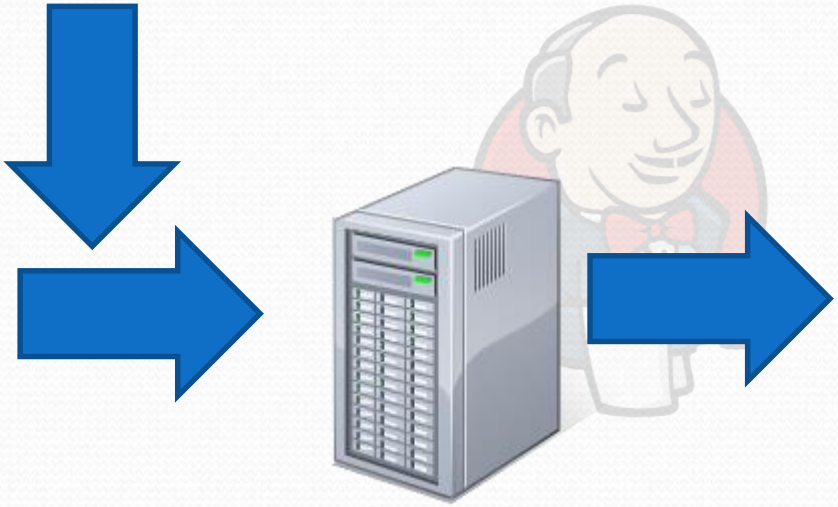
Решение: шаг 3

Screen2

Screen1 DataSet1 DataSet2 DataSet3

...

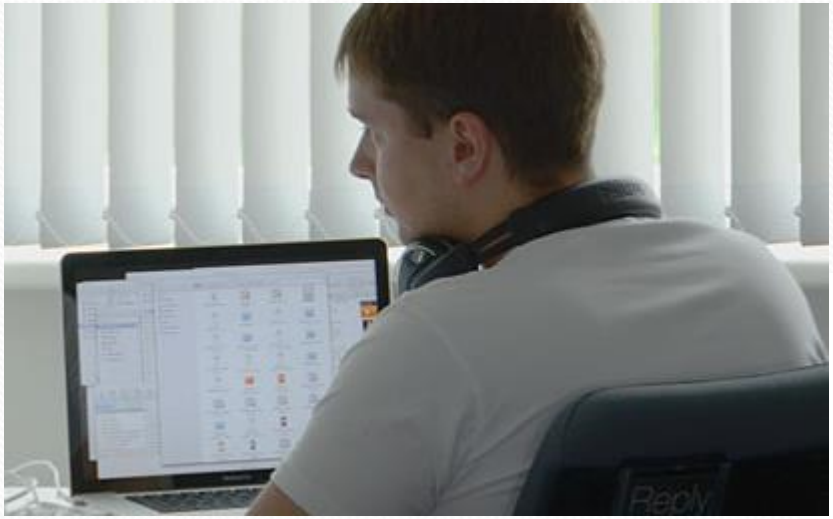
- 320x240
- 640x480
- 800x480
- 1280x720
- 1260x800
- 240x320
- 480x640
- 480x800
- 720x1280
- 800x1260
- ...



CI сервер

. . . .

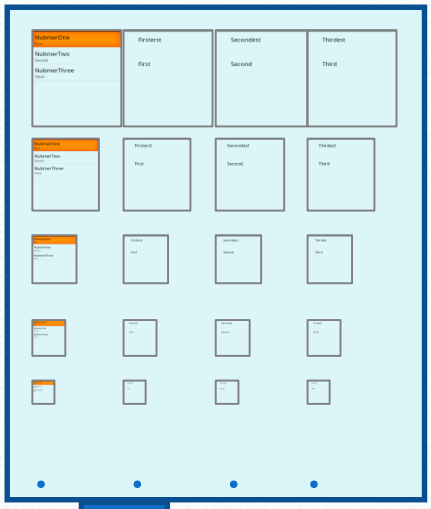
Первая проверка



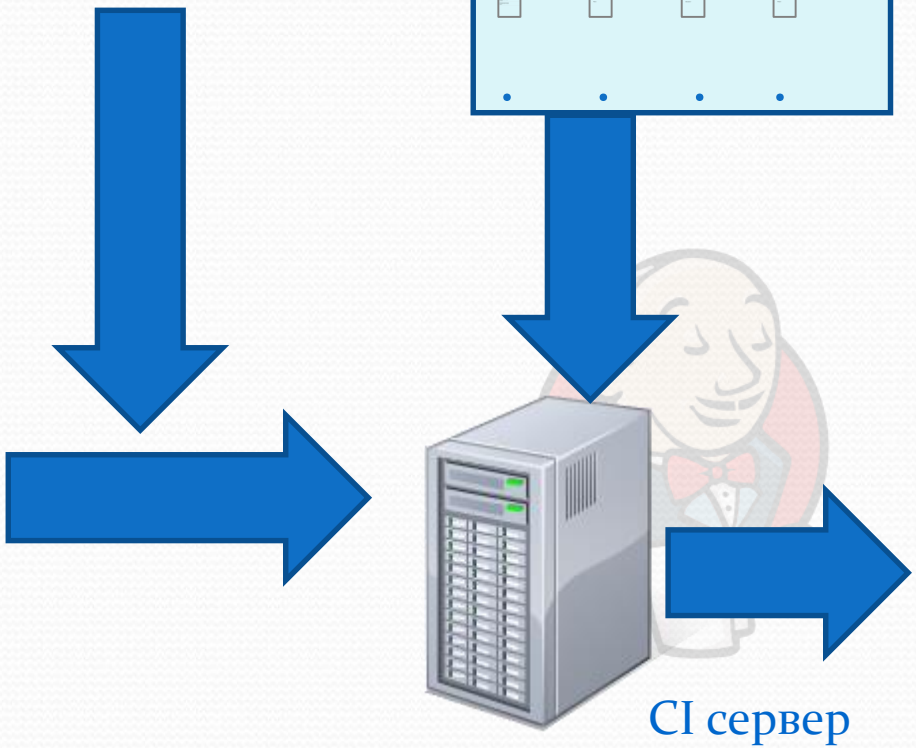
Решение: шаг 4

Screen1	DataSet1	Screen2 DataSet2	DataSet3
NubmerOne NubmerTwo NubmerThree	Firsterst First	Secondest Second	Thirdest Third

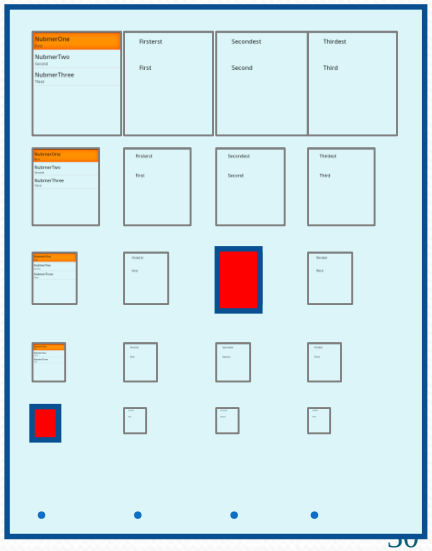
...



- 320x240
- 640x480
- 800x480
- 1280x720
- 1260x800
- 240x320
- 480x640
- 480x800
- 720x1280
- 800x1260
- ...



CI сервер



Последующие проверки

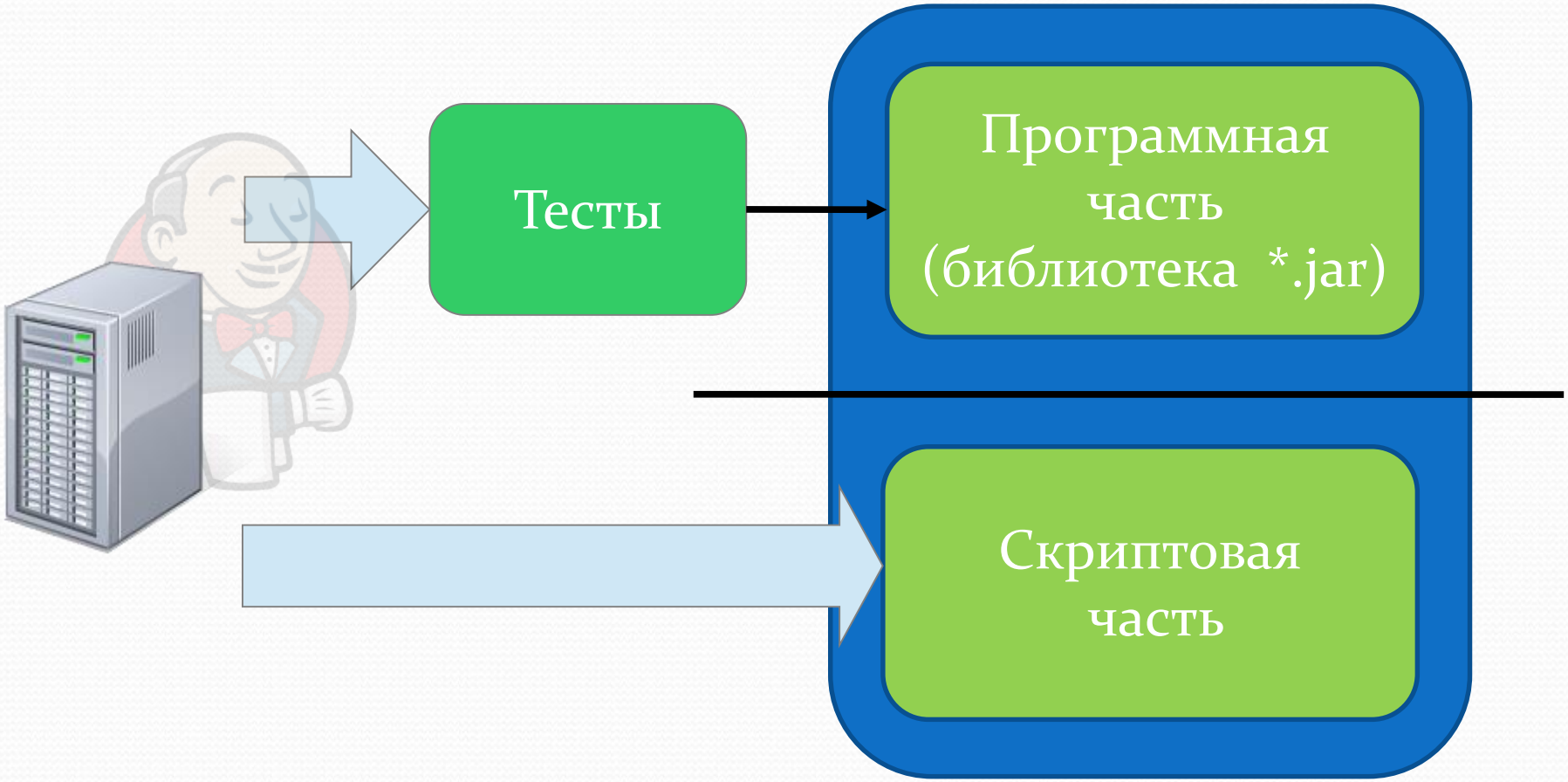
?

Что нужно сделать, чтобы это заработало

- Подготовить данные (БД или другое хранилище)
- Отвязать UI приложения от сервисов
- Сделать скриншоты экранов
- Сравнить два изображения и найти разницу

Реализация фреймворка

Состав фреймворка



Виды тестов в Android

Система Android на данный момент позволяет проводить два вида тестирования *Activity*:

- Unit-testing — тестирование, изолированное от инфраструктуры системы, позволет заменять некоторые объекты системы. Используется класс *ActivityUnitTestCase*
- Instrumentational-testing — тестирование, использующее реальную инфраструктуру системы, позволяет управлять действиями тестируемой *Activity* напрямую. Используется класс *ActivityInstrumentationTestCase2*

Состав библиотеки

Основные классы:

- ScreenshotTaker – непосредственно занимается созданием скриншотов

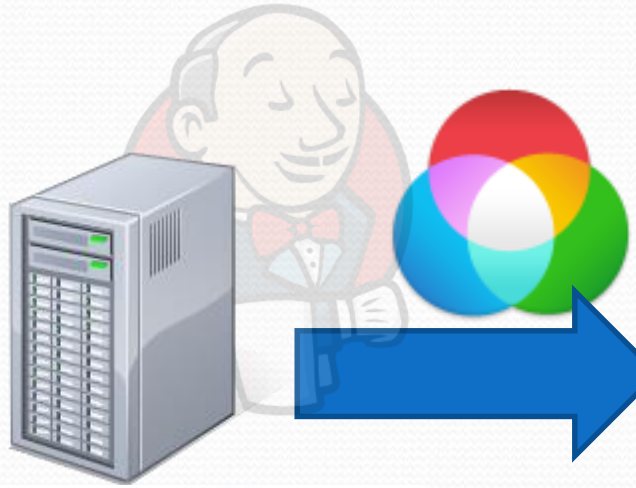
Вспомоготельные классы:

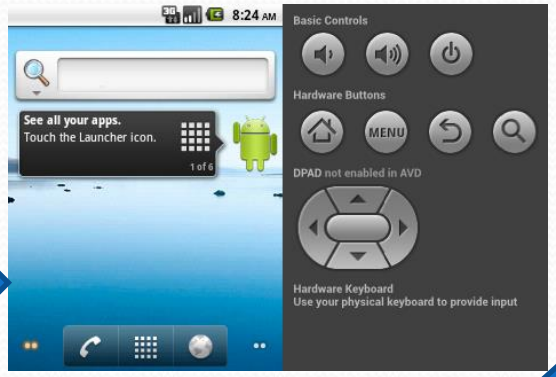
- DatabaseHelper – обнуляет базу данных и заполняет начальными данными

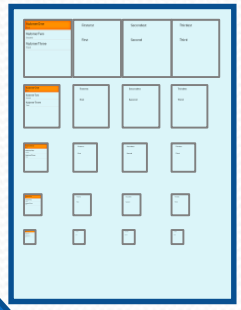
- DataSetReader – считывает исходные данные из файла

- Table – обеспечивает промежуточное хранение данных

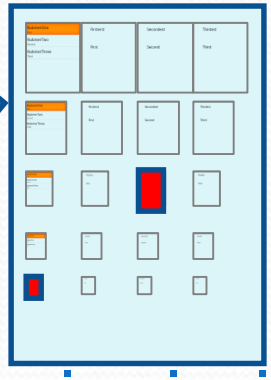
Схема работы теста







compare



Инициализация теста

- Инициализируем ScreenshotTaker

- Загружаем данные (если они одинаковые для всех тестов группы)

```
@Override
protected void setUp() throws Exception
{
    taker = new ScreenshotTaker(ScreenshotTaker.SavePathFormat.SUFFIX);
    activity = getActivity();
}

@Override
protected void tearDown() throws Exception
{
    activity.finish();
    dbHelper.close();
}
```

Загрузка данных в БД приложения

- Использовать DBUnit было сложно
- Разработали свой модуль загрузки данных
- Поддерживает один из двух форматов DBUnit

```
<?xml version="1.0" encoding="UTF-8"?>
<dataset>
    <table name="contacts">
        <column>first_name</column>
        <column>last_name</column>
        <column>display_name</column>
        <row>
            <value>Ivan</value>
            <value>Ivanov</value>
            <value>+7(913)123-45-67</value>
        </row>
        <row>
            <value>Second</value>
            <value>Secondest</value>
            <value>NubmerTwo</value>
        </row>
        <row>
            <value>Third</value>
            <value>Thirdest</value>
            <value>NubmerThree</value>
        </row>
    </table>
</dataset>
```

Загрузка данных в БД приложения

```
private void loadData(String datasetName) throws IOException
{
    dbHelper = new ContactsDbHelper(
            getInstrumentation().getTargetContext(),
            DbStruct.CONTACTS_DB_NAME,
            DbStruct.CONTACTS_DB_VERSION);
    InputStream file = getInstrumentation()
            .getContext()
            .getAssets()
            .open(datasetName);
    new DatabaseHelper(dbHelper.getWritableDatabase())
            .fillDatabase(file);
}
```

Выполнение скриншота

- Выполнение скриншота делается внутренними механизмами Android

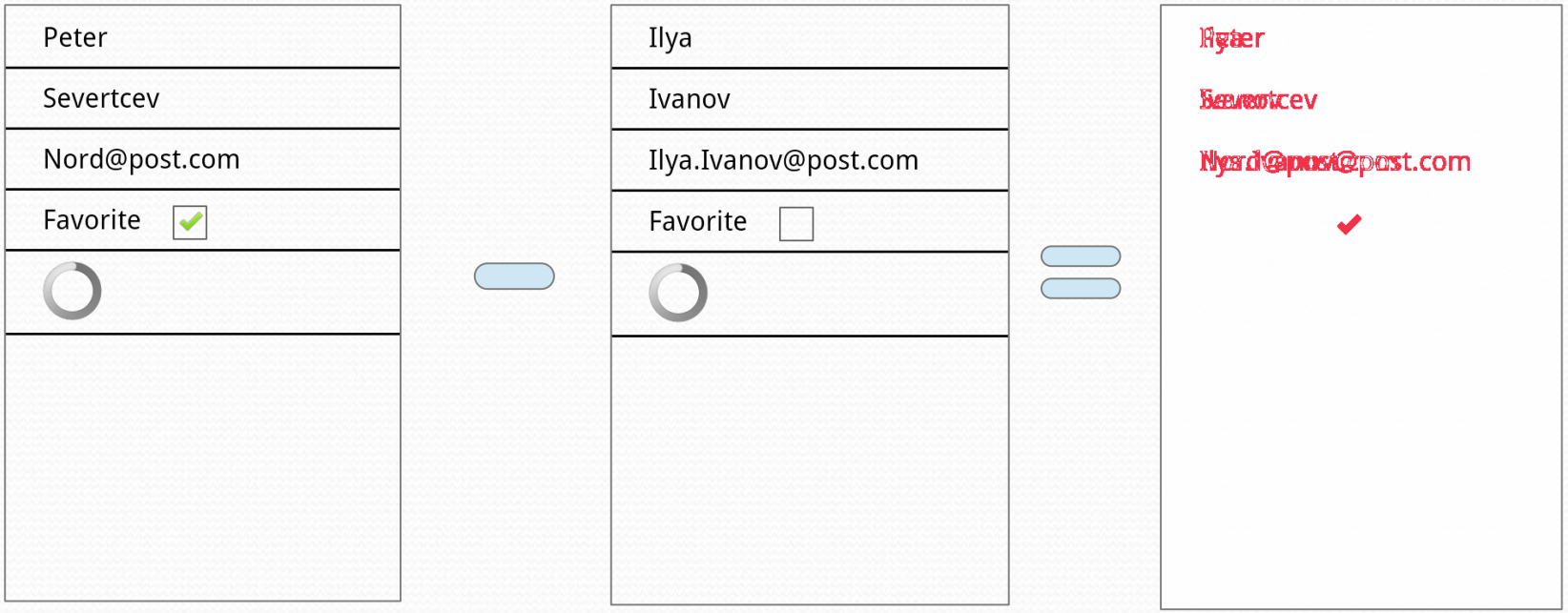
- Рассматривался вариант выполнения скриншота хостовой ОС

```
public void testContactsListScreenshot() throws IOException
{
    loadData("dataset.xml");
    taker.doScreenShot(activity, null, null, "contactListData");
}
```

Сравнение скриншотов

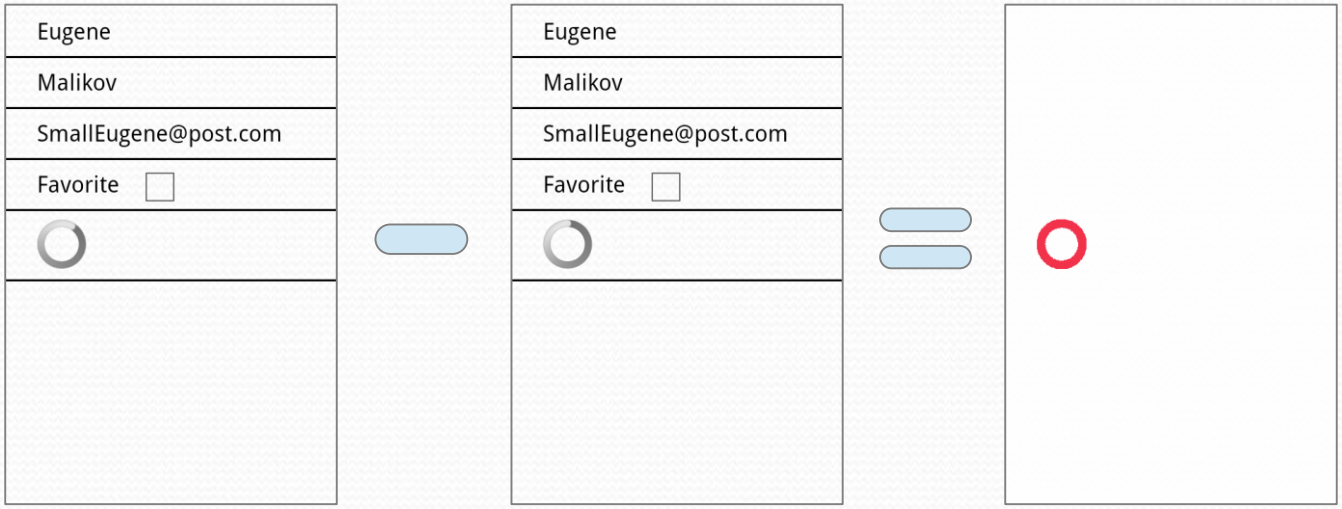
- Сравнение двух изображений построено на базе ImageMagick

- Различия между двумя скриншотами помечаются



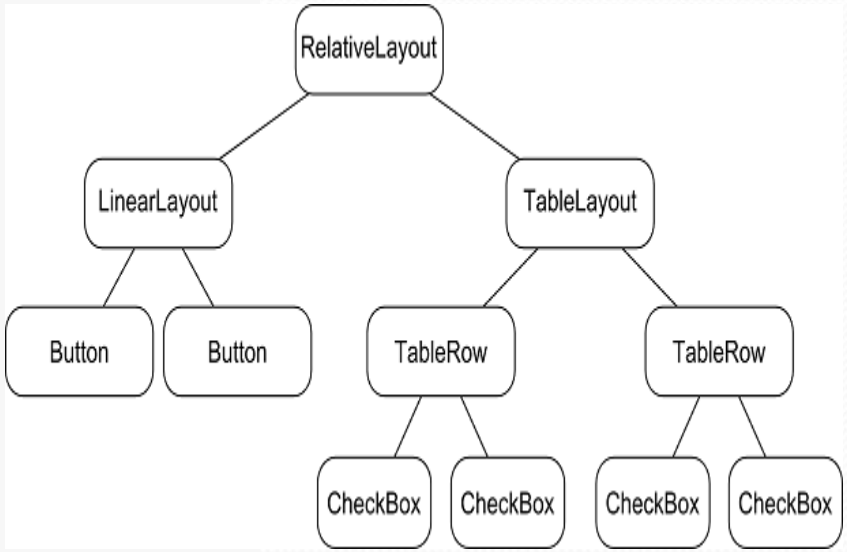
Исключение части экрана

Интерфейс иногда содержит компоненты, значения которых постоянно меняются, типа часов, прогресс баров и т.д.



Строение экрана в Android

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              android:layout_width="fill_parent"
              android:layout_height="fill_parent"
              android:orientation="vertical" >
    <TextView android:id="@+id/text"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:text="Hello, I am a TextView" />
    <Button android:id="@+id/button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Hello, I am a Button" />
</LinearLayout>
```



Входные данные метода

Метод, создающий скриншоты, принимает следующие агрументы:

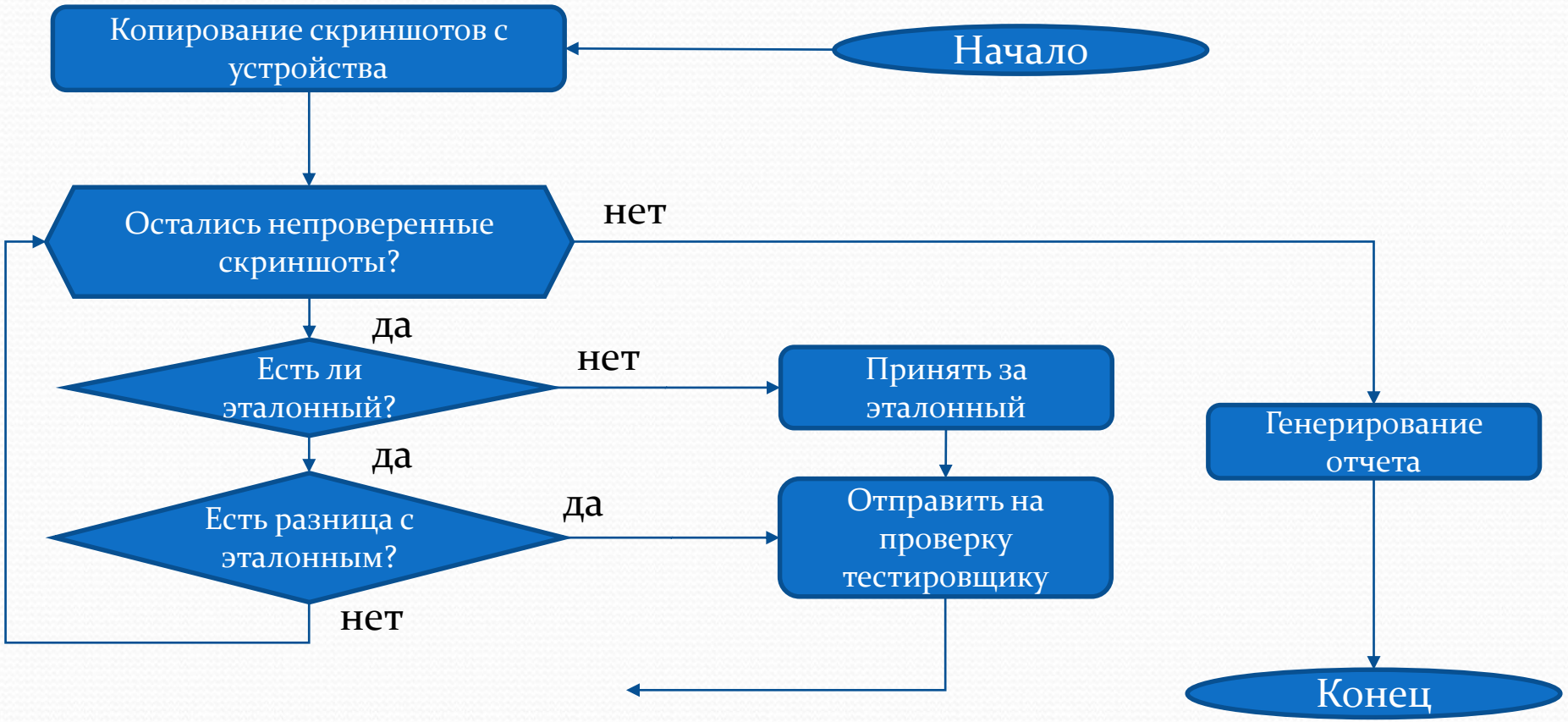
- Activity — та активити, скриншот элемента которой будет производиться
- Int — идентификатор элемента, снимок которого будет производиться
- Int — список индетификаторов тех элементов, котороые надо исключить
- String — уникальное имя скриншота, используется в именовании в файловой системе

Создание скриншота

Пример вызова функции, где создается скриншот корневого элемента с исключением элемента с идентификатором «*contact_progressbar*»

```
taker.doScreenShot(
        activity,
        null,
        new int[] {R.id.contact_progressbar},
        "contactData");
```

Алгоритм скрипта сравнения скриншотов



Заключение

- Был реализован фреймворк для ОС Android, позволяющий тестировать графический интерфейс приложения
- Планируется реализовать аналогичную систему для iOS, затем для WinPhone8
- Исходный код доступен по лицензии Creative Commons Attribution-NonCommercial-ShareAlike 3.0

Благодарим за внимание

https://github.com/tomsksoft/screen-tester-android

Сергей Борисов, tech lead, ТомскСофт
bsi@tomsksoft.com

Иван Лебедев, интерн, ТомскСофт
lia@tomsksoft.com

ТомскСофт
Программное обеспечение