<epam> Excellence in Software Engineering

Mobile Test Automation

Yet another Conference

October 1, 2012, Moscow

Test Automation vs. Mobile Test Automation

Software Test Automation provides **real benefits** at large, complex and long term projects.

It replaces entire teams of manual testers and save time of testing process.

But it is **not** about Mobile projects:

- they are short term
- not so huge
- don't have large manual QA team
- mobile technologies changes too fast – automation need continuous support

EPAM successful experience

EPAM has **successful experience** with Mobile Test Automation – it was successfully delivered to several mobile projects (**iOS** and **Android**)

- Mobile Test Automation was **fully integrated** into development and testing process
- Autotests were run **automatically** using **Continuous Integration** system with providing reports
- Tests can be run in **parallel mode** against different devices with different parameters (screen resolution, etc.) at the same time

Test Automation Tools

EPAM has experience of using the following tools for **Android** and **iOS**:

- Autotests for Android were implemented using **Robotiom** and **MonkeyRunner** (or their extended combitation)
- Autotests for iOS were implemented using **PhoneMonkey** and **UIAutomation**

Automation Tools evaluation

Before start of using **MTA** at real projects we conduct detailed **evaluation** of different test automation tools both for **Android** and **iOS**:

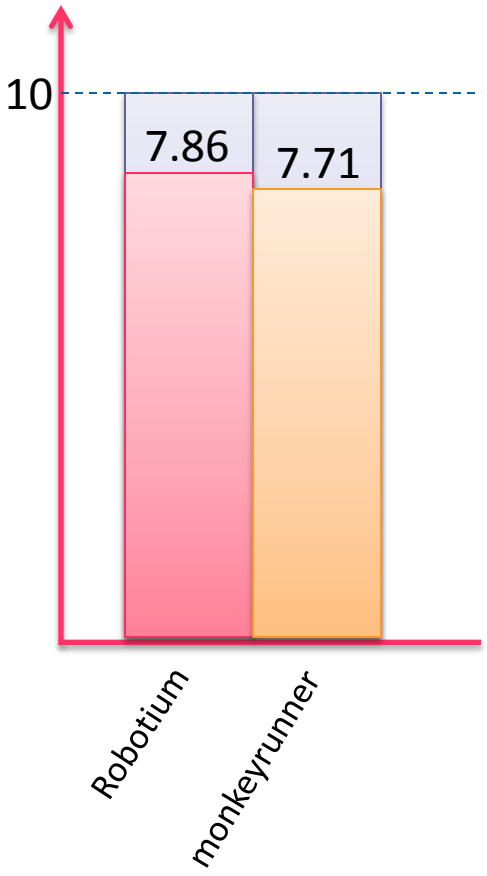
- Android: the evaluation was done for tne following tools: **Robotium**, **monkeyrunner**, **SeeTest**
- iOS: **PhoneMonkey**, **UIAutomation**, **UISpec**

Evaluation report

Example of the evaluation report

N	Evaluation area	Robotium	Rank	Notes
1	Record and Playback	Record not supported, good playback	4.86	4.86
2	Object identification	Practically full object identification supported	6.50	6.7
3	Verification/Control Points	High support of verification and control points	8.00	8
4	Test Data handling	Full support	9.80	9.8;5;7
5	Reporting	Full support	10.00	9;10;10
6	Scripting	Rather high support	8.14	7;7;7
7	Environment	Very good environment	9.17	9;8.5;7
8	Extensibility	Not so good extensibility	4.25	6;6
9	Licensing/Support	Open source, good future	10.00	10;9;9
		TOTAL	**7.86**	

N	Evaluation area	monkeyrunner	Rank	Notes
1	Record and Playback	Good record and playback	7.71	3;3
2	Object identification	Good supported, identification by images and IDs	8.00	1;2;2
3	Verification/Control Points	Rather high support of verification and control points	8.00	4
4	Test Data handling	Full support	9.80	9;10
5	Reporting	Full support	10.00	10;10
6	Scripting	Not so good support	7.71	5;6;6
7	Environment	Rather good environment	8.83	8
8	Extensibility	Not so good extensibility	4.25	6
9	Licensing/Support	Open source, good future	9.00	6;6
		TOTAL	**7.71**	



EPAM Mobile Project - Cost Tracking Center

Customer

EPAM Systems

Cost Tracking Center provides ability to create and to report about different kinds of payments, which can be compensated by the company: expense reports, business trips, purchase order, Invoices and etc.

Project Specifics

CTC is inner EPAM project, which is constantly developed and supported, grows and gets many new features.
Specifics of mobile version:

- Allows to control records in CTC remotely
- Allows to add any photo or scan of the document from the phone directly to the record
- Automation on Android

MTA Specifics

Initially CTC was created as WEB-Application. Then CTC migrated to iPhone and Android.

- Robotium was used for Test Automation to reduce the testing time

Robotium – Android Test Automation

Robotium is UI Test Automation framework for **Android**

Its **benefits** and **main features**:

- opensourse
- use Java and JUnit
- work with Android elements natively
- can run tests in parallel mode against several devices
- project is developing rapidly and support all Android versions
- Robotium Autotest is an Android application (Instrumentation) which is installed into the device and run there
- source code of the application under test is not needed

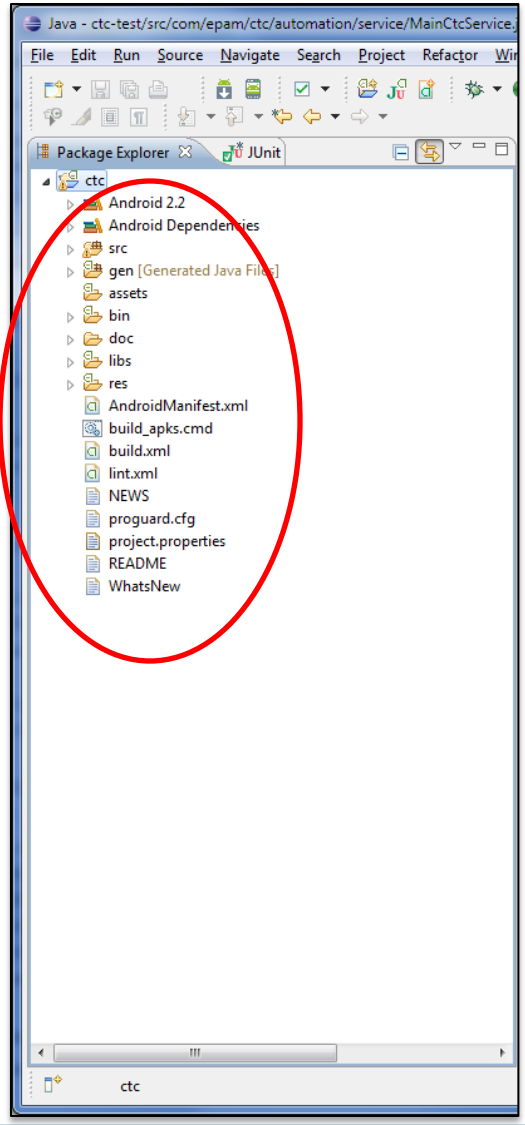
Robotium How To: creating simple autotest

Things, we need to create autotests with Robotium:

- **robotium-solo-xxx.jar** – actual version of Robotim library, can be downloaded from Google code
- **Eclipse IDE** with Android Development Tools (Android SDK also should be installed to the system)
- **Source code** of the application under test (AUT), or *.apk-file, but the signature should be the same. Or we can use android debug key

Robotium How To: creating simple autotest

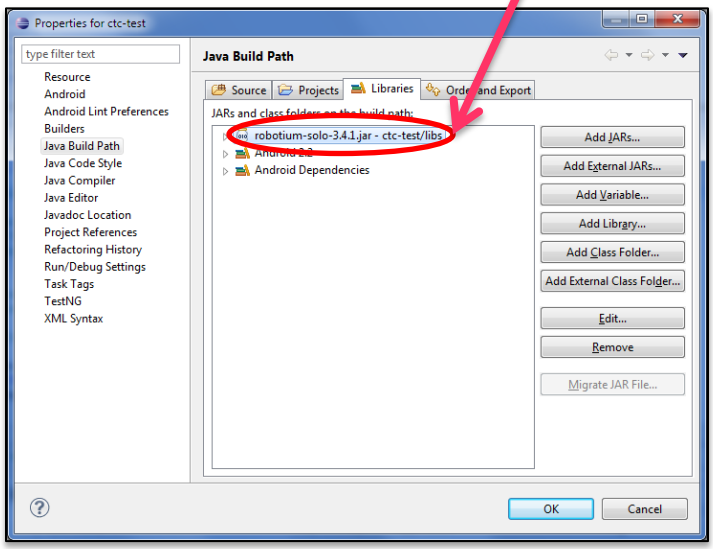
1. Application Under Test: Add source code of the application under test to Eclipse Workspace

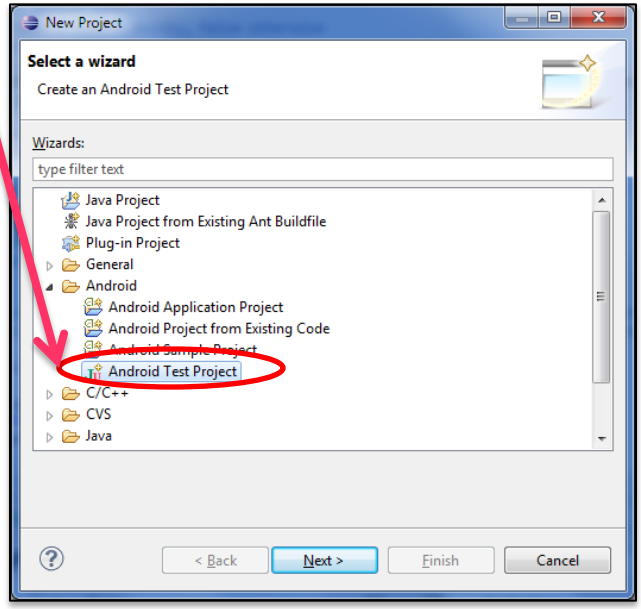


Robotium How To: creating simple autotest

2. Create new **Android Test Project** using Eclipse wizard. In a "*Choose a project to test*" category specify already existing AUT project

3. Go to just created project – "*Properties*" – "*Java build path*" – "*Libraries*". Add downloaded **robotium-solo-xxx.jar** to the project libraries

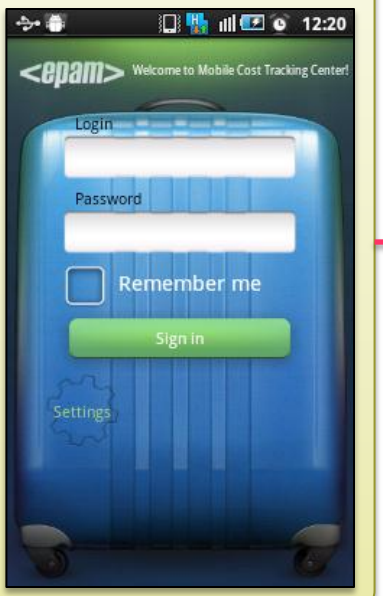




Robotium How To: creating simple autotest

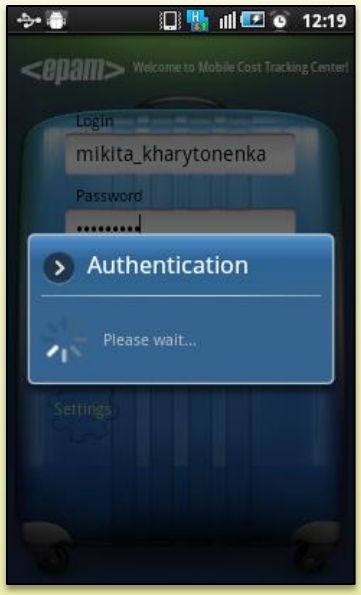
4. **Test case**: in our test we will login to application and check, that we done it successfully

1. Start application



2. Enter credentials

3. Press “Sign in”



4. Check, that welcome page is opened

Robotium How To: Creating simple autotest

5. Create **test class** for our first test like this

```
public class CtcSimpleTest extends ActivityInstrumentationTestCase2<MainView> {

    private Solo solo;

    public CtcSimpleTest() {
        super(MainView.class);
    }

    @Override
    public void setUp() {
        solo = new Solo(getInstrumentation(), getActivity());
    }

    @Override
    public void tearDown() throws Exception {
        solo.finishOpenedActivities();
    }
}
```

Test class should extends from **ActivityInstrumentationTestCase2**

Class **Solo** contains all **Robotium** functionality – methods click, touch, drag, type, etc. – like class **DefaultSelenium** in Selenium

In the **Constructor** we should pass the class of the View, which starts first in our application

JUnit **setUp** and **tearDown** methods.
Should contain accordingly:
- creating of the Solo object
- closing of the all opened activities

Robotium How To: creating simple autotest

6. Create **test method** in the class

```
public void testLoginToCtc() {

    solo.typeText(1, "mikita_kharytonenka");
    solo.typeText(0, "345ERTdfg");

    solo.clickOnButton("Sign in");
    solo.waitForText("Mobile CTC", 1, 10000);
}
```

According to JUnit, name of the test method should start from “test”

We identify editboxes by their indexes (or numbers) on the screen – 1st editbox, 2nd editbox, etc. Also we can use other type of identification – EditText Android class

Buttons are identified by their labels. They also can de identified by index (number).

After pressing the button, we are waiting for text appear on the next screen.

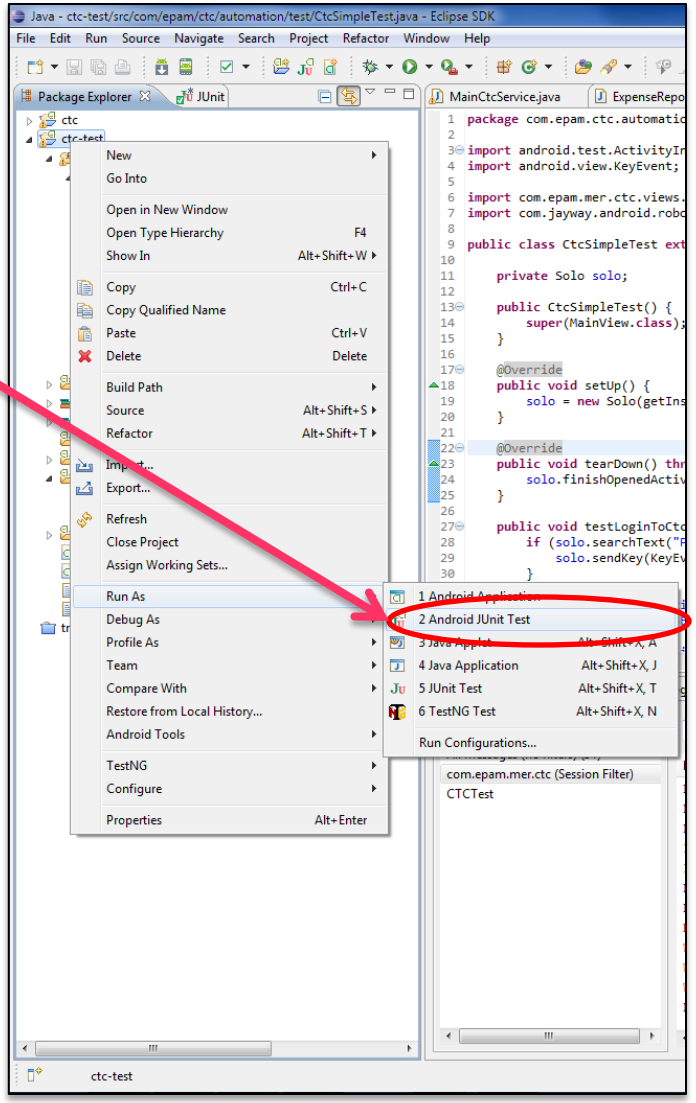
Robotium How To: creating simple autotest

7. **Running test**: test can be run against real device or emulator (it should be created first)

Robotium How To: creating simple autotest

To run test go to test project, choose “*Run as*” – “*Android JUnit Test*”

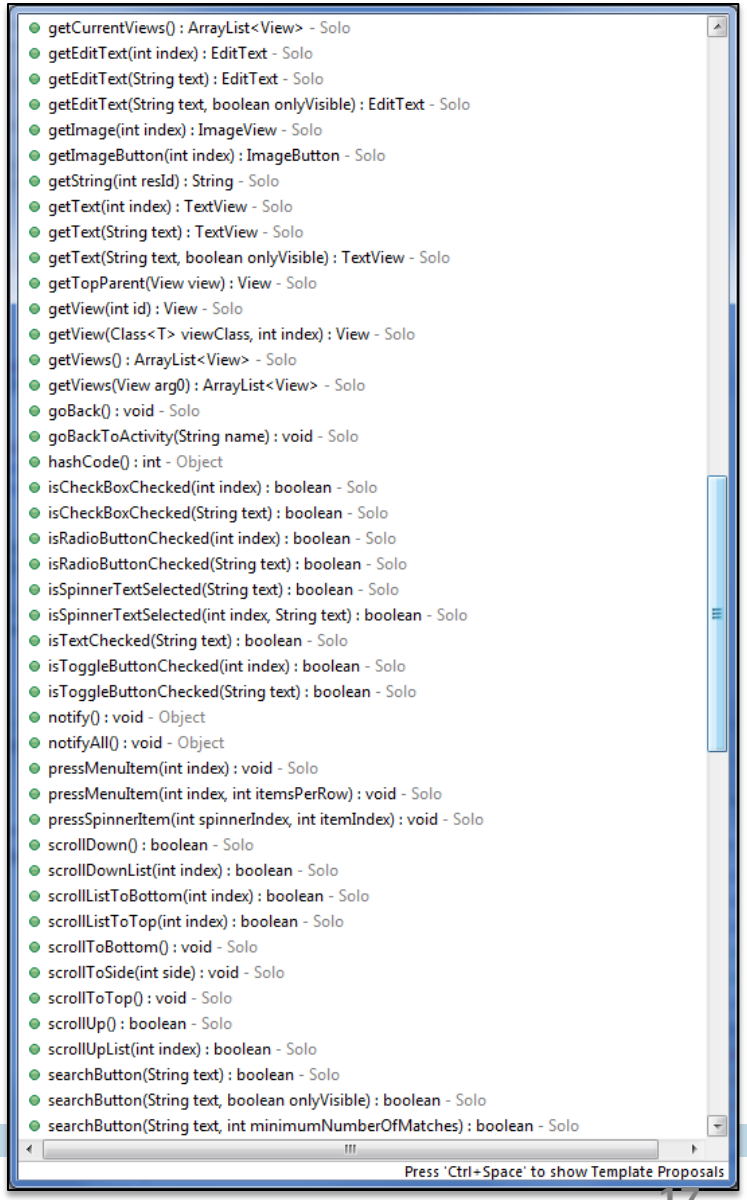
At the emulator or device you will see, how test performs



Robotium – class Solo

Class Solo provides us with great amount of methods, like

- scrolling
- typing
- touching
- dragging
- getting text and other properties
- ...



Robotium – Advanced features

Also Robotium has some interesting features, which allow to do our test more flexible and convenient for analyzing results

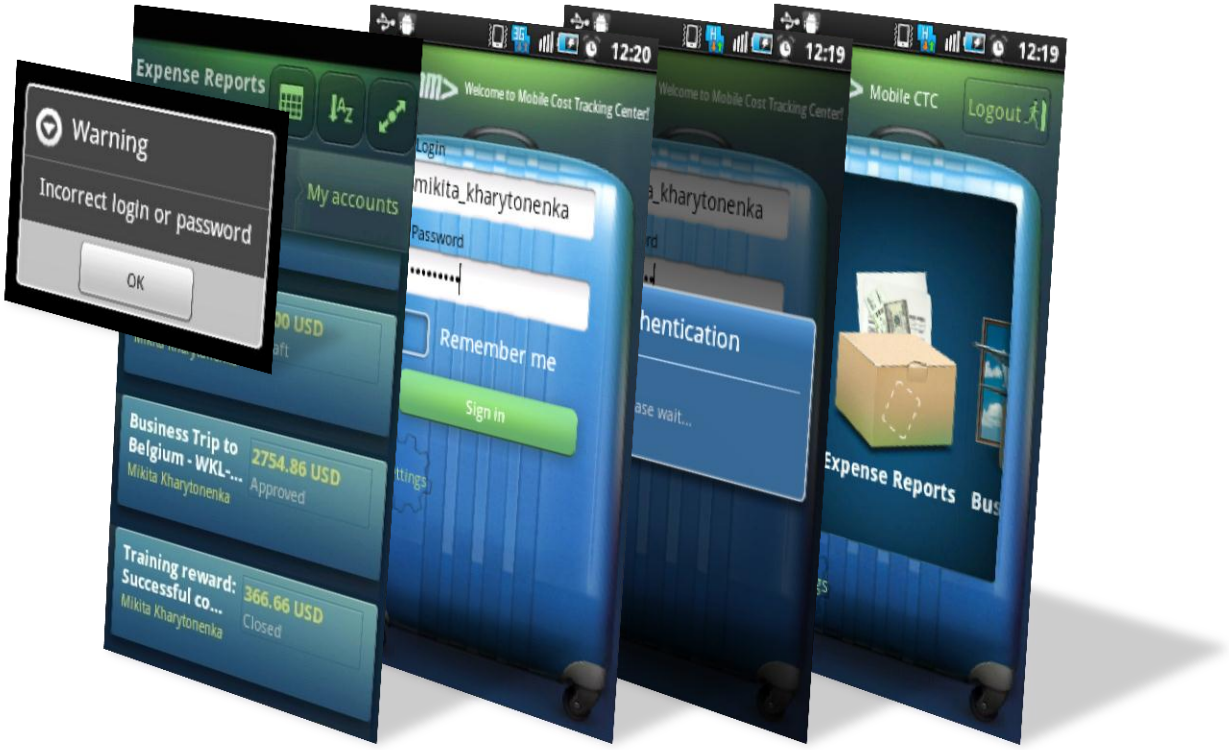
When we use emulator, we can **set the orientation** and test both landscape and portrait mode

```
solo.setActivityOrientation(ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE);
```

Robotium – Advanced features

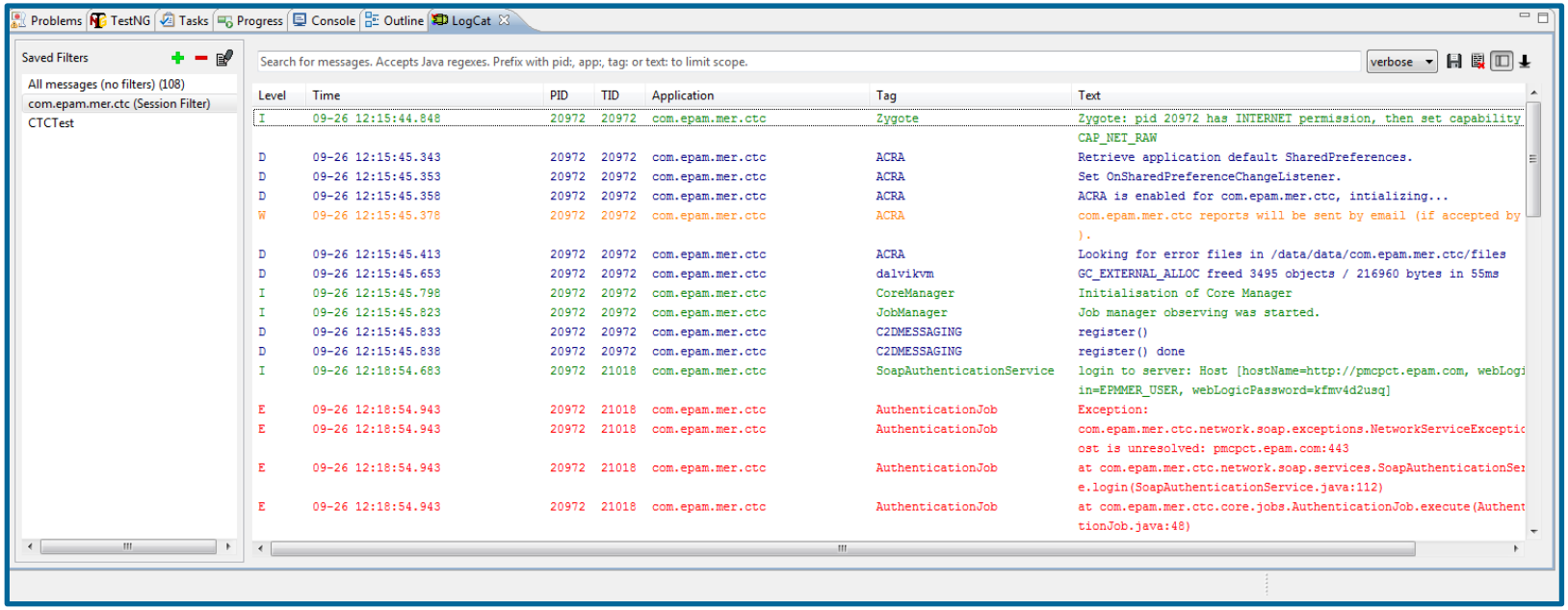
Robotium allows us to **take screenshots** anywhere in the test (both for Emulator and Device) and save it to device internal memory or SDCard or Emulator

```
solo.takeScreenshot();
```



Robotium – Advanced features

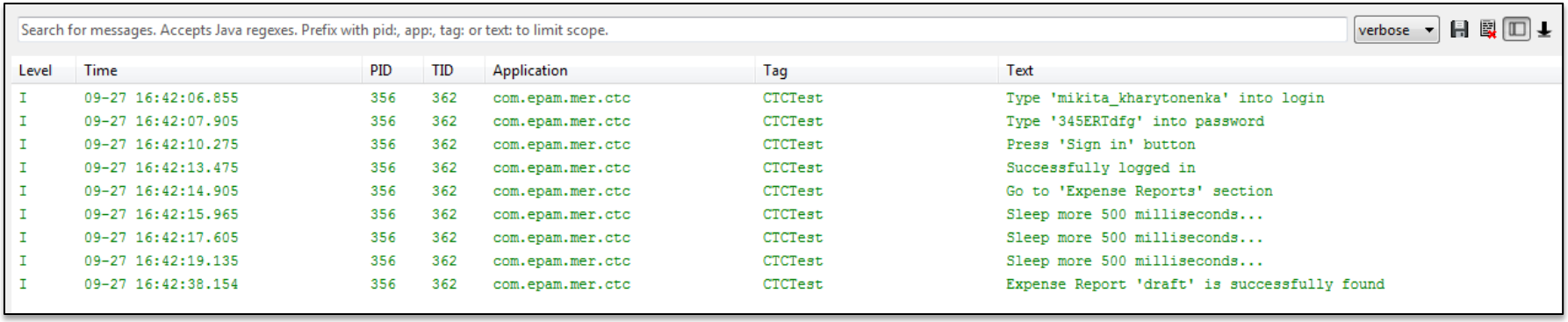
`solo.waitForLogMessage(String logMessage)` – waiting for message from **Android Logcat**, which can be performed by other application or etc. - one more function, allows to do test more flexible



Robotium – Advanced features

Logging different actions in the test:

```
Log.i("CTCTest", String.format("Type '%s' into login", userName));
```




```
09-27 16:42:06.855: I/CTCTest(356): Type 'mikita_kharytonenka' into login
09-27 16:42:07.905: I/CTCTest(356): Type '345ERTdfg' into password
09-27 16:42:10.275: I/CTCTest(356): Press 'Sign in' button
09-27 16:42:13.475: I/CTCTest(356): Successfully logged in
09-27 16:42:14.905: I/CTCTest(356): Go to 'Expense Reports' section
09-27 16:42:15.965: I/CTCTest(356): Sleep more 500 milliseconds...
09-27 16:42:17.605: I/CTCTest(356): Sleep more 500 milliseconds...
09-27 16:42:19.135: I/CTCTest(356): Sleep more 500 milliseconds...
09-27 16:42:38.154: I/CTCTest(356): Expense Report 'draft' is successfully found
```

Robotium – combination with monkeyrunner

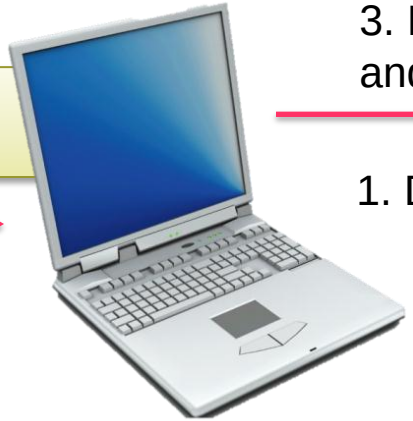
Main disadvantage of Robotium – it **can't work with different Applications** in on test – if your application call another one (like Camera) – Robotium cant "see" it and press any buttons there.
Our workaround – use monkeyrunner tool to perform this action – click on the Camera button to take picture.

How it works:

```
from com.android.monkeyrunner import MonkeyRunner, MonkeyDevice
device = MonkeyRunner.waitForConnection()
device.touch(15, 215, 'DOWN_AND_UP');
```

3. Monkeyrunner Python script is run by SSH command and performs at the device

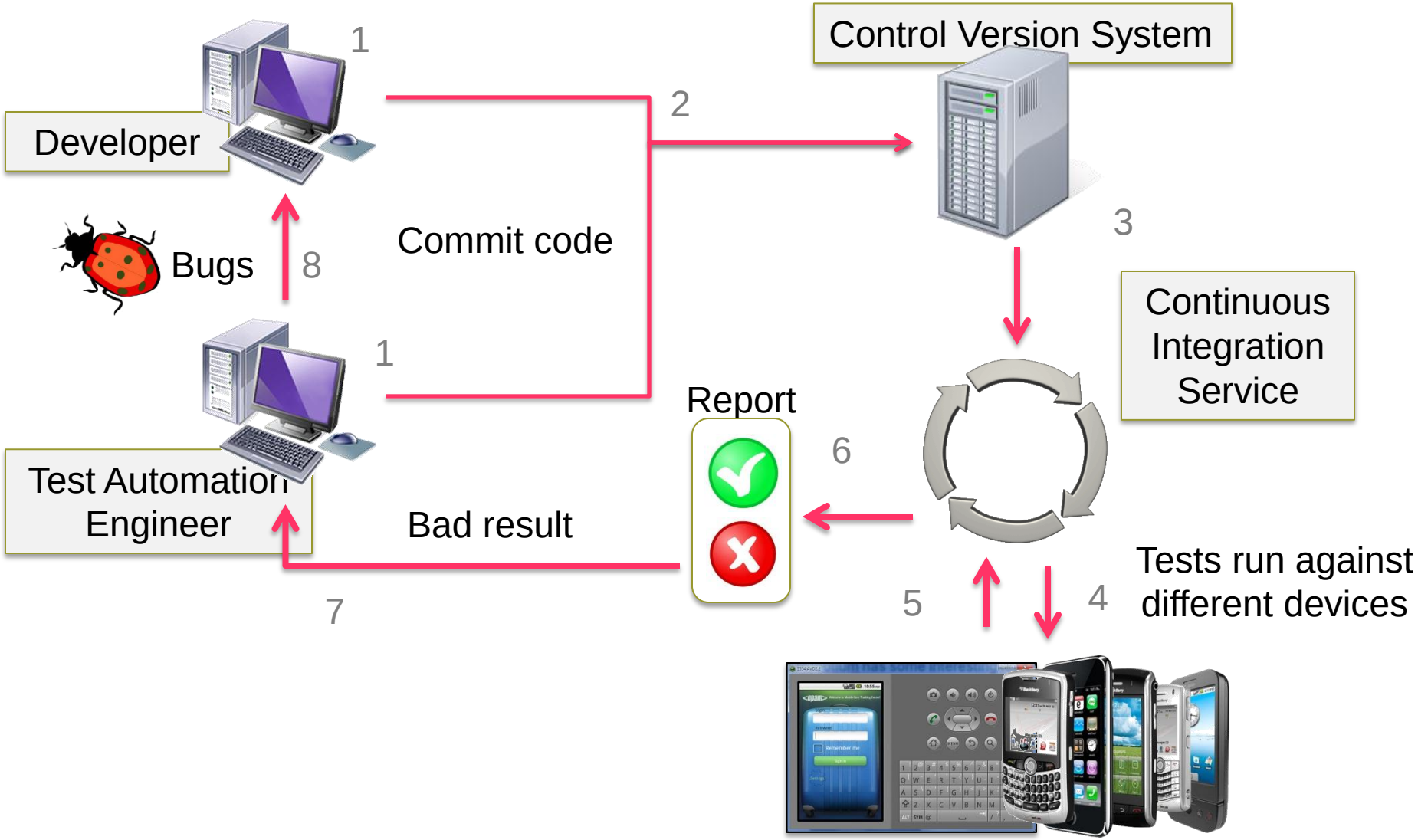
Unix machine or Windows machine with OpenSSH



1. Develop tests and upload it to device

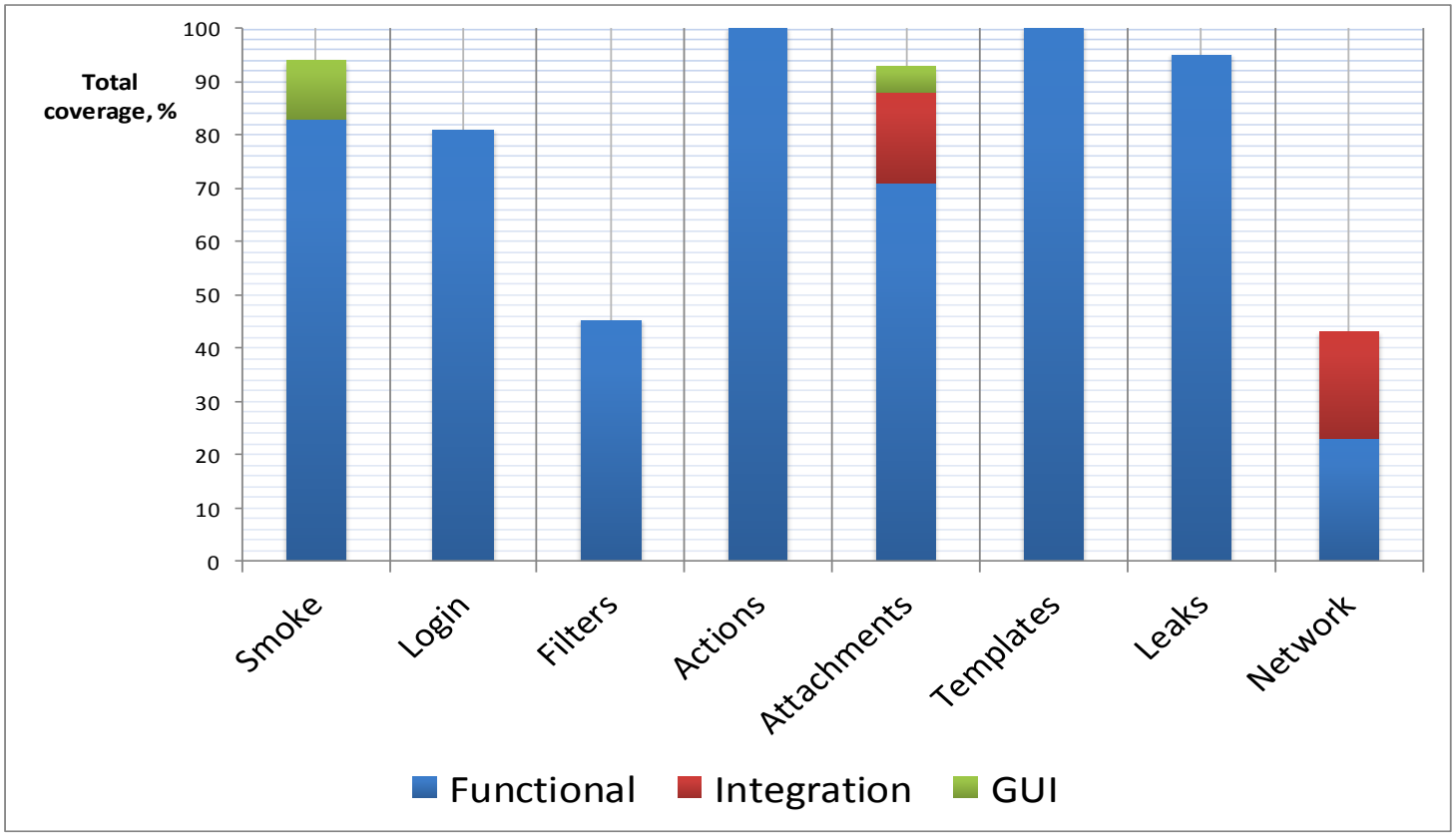
2. Test runs at the device and connect to machine via SSH

Robotium and Continuous Integration



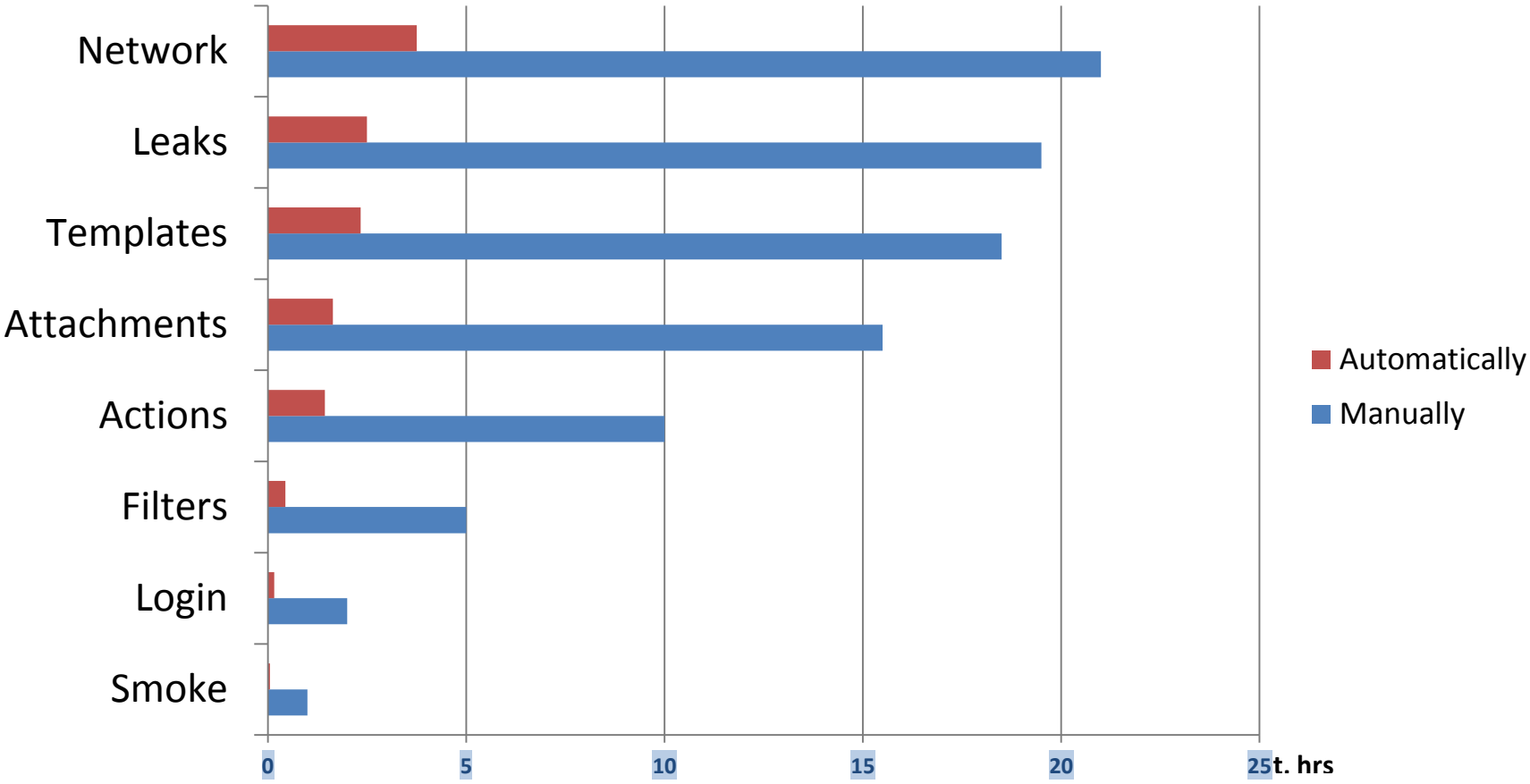
CTC – Automation Tests Coverage

Not all functionality were covered by autotests. Firstly, we analyzed test cases and decided, **what we will automate**.
There were several cases which we decided not to use automation due to manually we spend rally less time then creating autotests



Mobile Test Automation – Benefits

As for time saving – graph shows us how much time were saved comparing manual and automated test running



About Speaker

Mikita Kharytonenka

<epam>

Senior Software Test Automation Engineer

Email: mikita_kharytonenka@epam.com
Skype: mikita_kharytonenka