White Paper

Test Automation Tools for Mobile Applications: A brief survey

1.0 EXECUTIVE SUMMARY

While UI test automation is well understood for the desktop market, with a plethora of good tools available, the mobile market is different. Due to platform fragmentation and restrictions imposed by mobile device OSs, UI automation is a harder problem to solve. Over time, several tools have evolved that offer different mechanisms for UI automation for mobile devices, each with certain advantages and disadvantages.

The goal of this document is to compare and contrast some of the popular tools against various dimensions to help developers assess suitability of such tools for their need. The document limits itself only to provide information about the features / capabilities available in different tools and does not describe in detail on how to use that particular tool. Further, this document will only focus on Android and iOS when comparing such tools.

CONTENTS

SECTION

2.0 MOBILE TEST AUTOMATION CHALLENGES

Test automation is the use of software to automate and control the setting up of test preconditions, execution of tests, test control and test reporting functions, with minimal user intervention. It is not practical to try to automate everything, especially for mobile devices. However, leveraging Common Off The Shelf (COTS) tools can greatly benefit the automation process.

The following are some of the reasons that make effective testing automation of mobile applications challenging:

- A mobile device is much more restricted compared to desktops – the underlying OS typically sandboxes each application and allows very limited inter process access, unless a phone is 'rooted'.
- Controlling the UI navigation of a mobile device is harder – in addition to lack of control mentioned in the previous point, mobile UIX response times are harder to predict than desktop equivalents and hence makes screen grab based predictions of pass/fail harder.
- Mobile devices, by definition are not statically located entities – depending on where the device is, the network may introduce many challenges that completely break down a prior tested use-case
- The proliferation of different screen sizes and form factors make UI based testing more challenging as well

However, mobile test automation provides many advantages, including:

- Improves testing efficiency
- Consistent and repeatable testing process
- Improved regression tests
- More tests can be run in less time - Improved coverage in shorter time
- 24/7 operation - better resources utilization
- Human resources are free to perform advanced manual tests
- Simple reproduction of found defects

3.0 TYPES OFTEST AUTOMATION TOOLS

Figure 1 describes the typical phases in mobile application testing and feature enhancement. Most mobile applications today are developed in tandem with an Agile software development methodology which involves very frequent interim releases. In such cases, implementing test automation can greatly help reduce iterative time to market. To achieve test automation it is worthwhile to discuss various tools available and the nature of testing they enable that could be aligned to the an Agile software delivery process. This is discussed next.

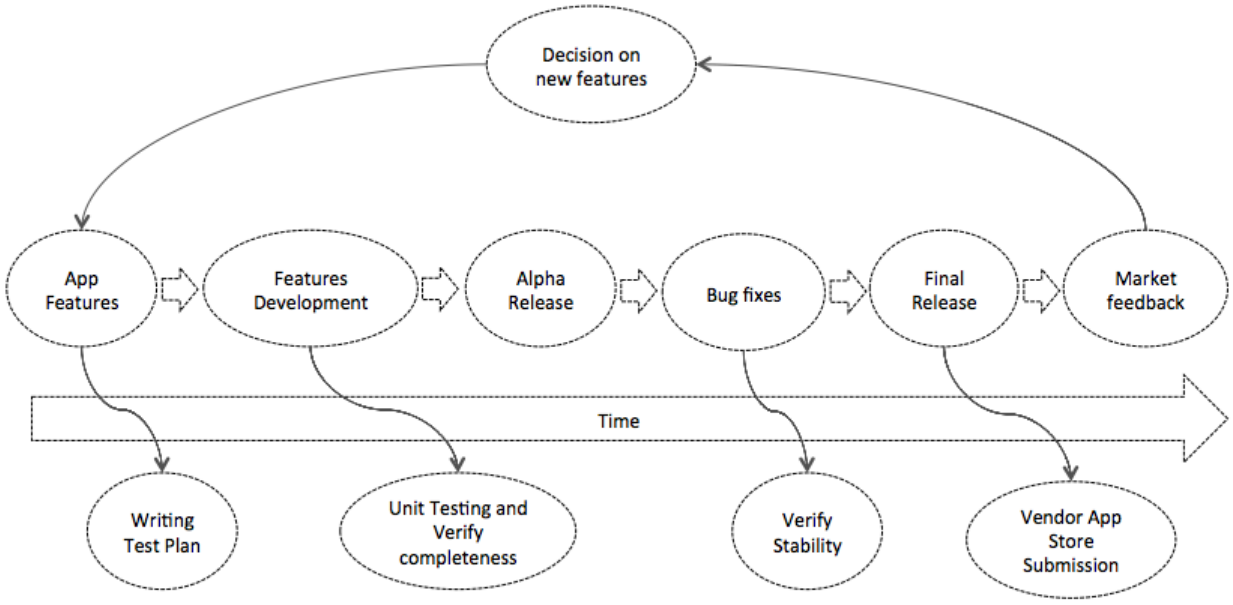


Figure 1Mobile Application Development

3.1 GUI Test Automation Tools for iOS and Android

There are quite a few GUI test automation tools available in the market. We have broadly categorized these tools as below:

1. **Platform Specific Tools** -. These are often provided by the Phone OS vendor (example, bundled by Google as part of the SDK, or by Apple as part of Xcode) and have integrated support with the SDK IDE. Platform specific tools are capable of testing the application both on the target device and on the emulator. The more popular platform specific tools are:

    - **Instrument**[1]–Instrument is a Mac application provided by Apple as part of Xcode (IDE for IOS Application Development). A new feature called “Automation” was added in Xcode 4(onwards) to automate GUI testing of iOS Apps. The Automation instrument requires a basic understanding of JavaScript for test scripts development. Apple provides a set of JavaScript libraries that can be used to drive tests and simulate user interaction.

[1]http://bit.ly/fZDCKT

- **Monkeyrunner**[2] - monkeyrunner provides an API for writing programs that can control an Android device or emulator remotely. With monkeyrunner, you can write a Python program that installs an Android application or test package, run it, send keystrokes to it, take screenshots of its user interface, and store screenshots on the workstation. This provides a powerful hook to integrate monkey runner with other Desktop based test tools to control the device and compare the resulting screenshots with reference screenshots for automation purposes. The monkeyrunner tool is primarily designed to test applications and devices at the functional/framework level and for running unit test suites.

2. **Generic Script Based Tools**– These tools are typically developed by third parties and usually control the execution of the application using scripts. These scripts can be written in a text editor or it can be generated automatically by recording events from your application. Below are the list of few tools in this category. In general, generic script based tools often need to integrate with some type of platform specific tool to complete automation, where the platform specific tool is responsible for communicating with the device, while the generic script based tool interfaces with it via an adaptor (that either is provided by the third party, or needs to be developed). Some of the popular tools here are:

    - **Sikuli**[3]- Sikuli is a visual technology to automate and test graphical user interfaces (GUI) using images (screenshots). Sikuli includes Sikuli Script, a visual scripting API for Jython, and Sikuli IDE, an integrated development environment for writing visual scripts with screenshots easily. Sikuli Script automates anything you see on the screen without internal API support. You can programmatically control a web page, a Windows/Linux/Mac OS X desktop application, or even an iPhone or android application running in a simulator or via VNC.

    - **Robot Framework**[4]- Robot Framework is a generic test automation framework for acceptance testing and acceptance test-driven development (ATDD). It has a easy-to-use tabular test data syntax and utilizes a keyword-driven testing approach. Its testing capabilities can be extended by test libraries implemented either with Python or Java, and users can create new keywords from existing ones using the same syntax that is used for creating test cases

3. **Random Event Generator –** There are many testing tools available, which test the stability of the application by sending random events to it. These tools are not intended to run any particular test case but quite often use to test stability of the application.

    - **UI/Application Exerciser Monkey**[5] - Monkey (not to be confused with Monkeyrunner above) is a program that runs on an Android emulator on Android device and generates pseudo-random streams of user events such as clicks, touches, or gestures, as well as a number of system-level events. You can use Monkey to stress-test applications that you are developing, in a random yet repeatable manner.

    - **Automation** – By default iOS Automation template doesn't have any random event generator but one can easily develop it utilizing the JavaScript and iOS automation API's. Using automation JavaScript one can create custom modules which can provide all the capabilities like touches, gestures, system-level events and add the capabilities to test the application for custom designed tests suits.

4. **Whitebox testing tools –**Whitebox testing tools can examine the internal structure of the application via control flow testing, data flow testing, and code branch testing. For white box testing, the test

[2] http://developer.android.com/tools/help/monkeyrunner_concepts.html
[3] http://www.sikuli.org/
[4] http://code.google.com/p/robotframework/
[5] http://developer.android.com/tools/help/monkey.html

developer needs to have access to the source code for application under test, or at the least be able to link their test library to the Application Under Test (AUT) for a special build. Following are some of the tools that can be used for whitebox testing of mobile applications:

- **Android Instrumentation:** Android instrumentation test framework is based on JUnit. It allows JUnit to be used for testing of Classes/Method that does not use Android specific APIs. For testing of Android specific Classes/Methods Android extensions of JUnit are used.
- **Instruments:** Instruments tool is part of the iOS application development IDE and allows Whitebox testing of iOS applications. Frank and TestStudio tools also allow Whitebox testing.

5. **Blackbox testing tools –**Blackbox testing tools check the functionality of the application and not how that functionality is achieved. Blackbox test tools do not require the test developer to have access to application code or to know how the code is organized or how it internally works. Following are some of the tools that can be used for Blackbox testing of mobile applications:

    - **Sikuli:** As explained earlier, Sikuli uses a visual technology for test automation and does not requires test developer to have any knowledge of the application code**.**
    - **Robotium[6]:** Robotium is a test framework for Android that allows test developers to create blackbox as well as white box test cases for Android applications.

    Subsequent chapters will describe the applicability of some of these tools in more detail.

[66] http://code.google.com/p/robotium/

4.0 SIKULI

4.1 Introduction

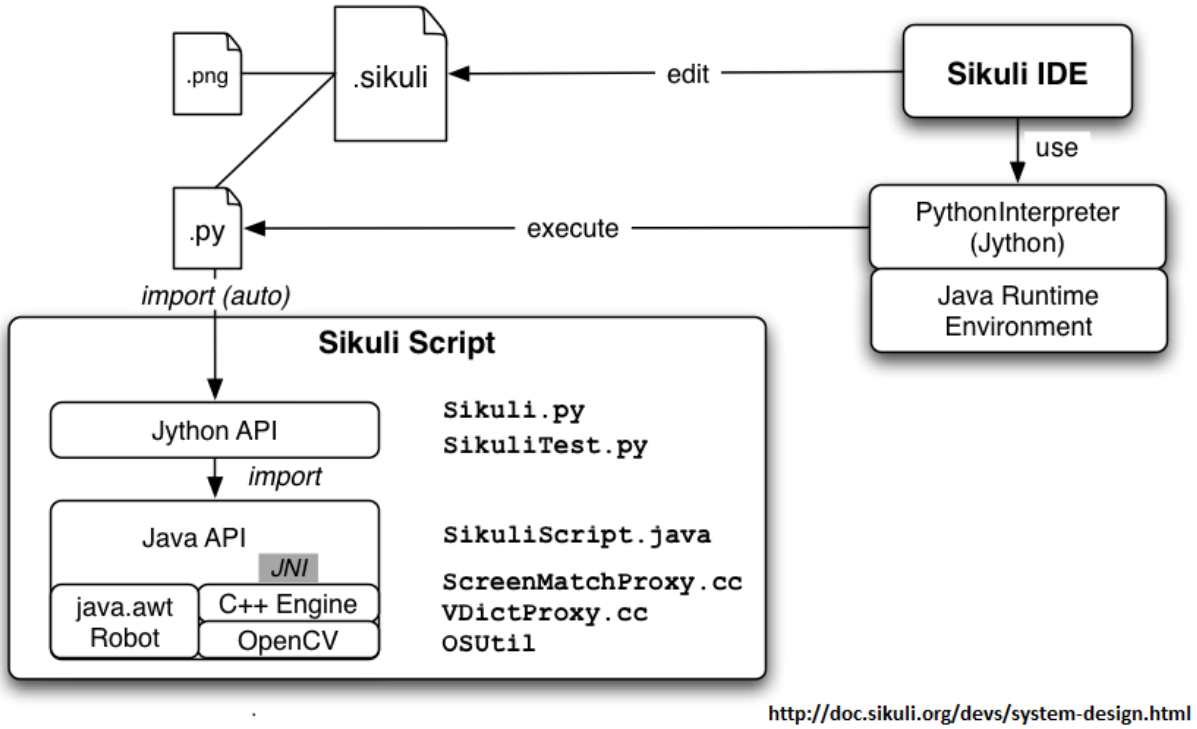
Sikuli is a visual technology tool to automate testing of graphical user interfaces (GUI) using image recognition. [7]Sikuli scripts can be used to take a snapshot of a GUI element and compare it with some reference snapshot. Since Sikuli uses a visually driven approach (i.e. drag/drop images to test against), a test script writer can write test cases without any knowledge of the target application code. Sikuli provides an IDE that allows for rapid development of test scripts with application screen shots. It allows automation of anything that is visible on a screen. Sikuli can be used to control a web page, a desktop application, an android application or even an iPhone application running in a Simulator.

An application that satisfies the following criteria can be tested with Sikuli:

1. Application screen can be transferred to a desktop (Windows/Mac/Linux) where Sikuli is being run.
2. User input to application can be transferred from desktop to application

Sikuli scripts are Jython scripts and can be extended to use other available Jython modules.

4.2 How Sikuli works



Sikuli uses images patterns to deliver mouse/keyboard events to appropriate GUI elements. Sikuli consists of two main parts

1. Java.awt.Rotot: This component of Sikuli delivers user events to the desired location on screen.
2. C++ Engine: Sikuli's core image processing engine is based on OpenCV (Open Source Computer Vision). This component is responsible for searching image patterns on screen.

4.3 Sikuli for Mobile Platforms

[7]http://www.sikuli.org/

Sikuli runs on Windows/Linux/Mac platforms. To be able to test mobile applications with Sikuli, the mobile phone/tablet screen should be transferred to Desktops where Sikuli is installed. Following are some of the ways to test mobile applications with Sikuli:

4.3.1 iOS application testing with Sikuli

One of the possible way to test an iOS application with Sikuli is to run the iOS application in an iPhone/iPad simulator within Xcode. The other possibility is to run a VNC server on the iOS device and run a VNC client on the desktop to get the iPhone/iPad screen on the Desktop. This approach has a disadvantage that it can only be used with a jail broken iOS device because VNC servers are not available for non-jail broken devices.

4.3.2 Android application testing with Sikuli

For testing Android applications there are two possibilities to transfer the mobile phone/tablet screen to the desktop.

- Screen Cast application which uses the Android Debug Bridge
- VNC server

The Screen cast application can transfer an Android phone/tablet screen to the desktop. However, it requires the phone to be rooted to be able to pass keyboard/touch events to the phone. Most of the VNC servers also require rooted Android phones. One VNC server, “VNCLite” available at Google Play does not require a rooted phone and works well with Sikuli.

Note: For ICS and later releases, Screen cast application cannot pass type/touch event to phone even for rooted phones.

4.4 Sample Script

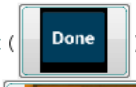
Below is a sample Sikuli test script. This test script automates the test case to add a news category and RSS news feed for Hughes Systique Pace [8]application. Following is the list of activities performed by the test script

1. Click on Home button so that any running application is put in background
2. Click on application tray button and then launch “HSC Pace” Application
3. Wait for splash screen to finish
4. Click on button to add news category
5. Type the category name
6. Check for successful addition of category
7. Click on button to add RSS feed and then type RSS URL
8. Check for successful addition of URL

[8]https://play.google.com/store/apps/details?id=com.hsc.android.pace

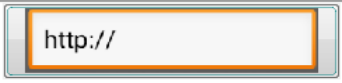
```
//Click on Application tray button
click(  )
//Clcick on Launcher for HSC Pace Application
click( 

)
//Wait for Splash Screen to finish
waitVanish(  ,30)
//Wait for button to add New feed category
wait(  ,30)
wait(2)
click(  )
//Click in text box
click( 

)
//Type news category name in text box
type("TestCategory")
//Click on done button of keyboard
click( 

)
wait(  ,30);
//Click on button to add RSS feed
click(  )
//Wait for plus button to apper
wait(  ,10)
//Click on plus button
click(  )

wait( 

,10)
click( 

)
//Type RSS URL
type("www.engadget.com/rss.xml")
click( 

)
wait(5)
//Wait for successfull addition of RSS feed
wait( 

,50)
```

4.5 Pros

1. Does not require any knowledge of the application code/design
2. Easy and quick to automate
3. Scripts are not dependent on underlying platform
4. Can be used for testing any GUI
5. Sikuli provides a 'fuzzy logic' image comparison that can be adjusted during the testing process to determine degree of match.

4.6 Cons

1. Its not easy to match images if their dimensions/locations change – Sikuli offers various primitives but depending on nature of change, matching logic may or may not work.
2. Not suitable for performance testing of mobile application because mobile phone screen transfer to Desktop using VNC/Screencast is not very smooth. This is actually a VNC/Screencast limitation rather than Sikuli limitation.

5.0 MONKEYRUNNER (FOR ANDROID)

5.1 Introduction

Monkeyrunner tool is part of AOSP (Android Open Source Project) which provides APIs for writing programs that can control an Android device remotely. With MonkeyRunner, we can write python programs that install an Android application, send keystrokes to it, take screenshots of its user interface, and store screenshots on the Test Driver. The screen shots can then be compared with sample baseline screenshots and checked for accuracy.

MonkeyRunner, in general, provides the facilities for functional testing, regression testing as well as ways to extend the tool and write frameworks on top of it. MonkeyRunner provides access to three basic classes MonkeyRunner, MonkeyDevice and MonkeyImage. These classes provide a set of APIs which could be invoked from custom Python programs to create and execute test cases.

The Plug-in architecture of MonkeyRunner also allows users to add their own classes to the API.

5.2 Setup and Installation

Monkey Runner does not need to be installed separately as it is part of Android SDK.

5.3 Sample Script

The sample script below demonstrates a simple test case for the following actions:

1. Install an Android Application
2. Open a UI screen
3. Click on a button to open another screen
4. Take the UI snapshot
5. Compare it with the reference snapshot

```
# Import Monkeyrunner Classes
fromcom.android.monkeyrunner import MonkeyRunner, MonkeyDevice
#This class is not documented
fromcom.android.monkeyrunner.easy import EasyMonkeyDevice

# Connect to Android Device
mydevice = MonkeyRunner.waitForConnection()

# Install the Android Application on connected device
device.installPackage('hscTest.apk')

# Package Name
package = 'com.hsc.TestApp'

# set  variable with Activity Name
activity = 'com.hsc.TestApp.LandingPage'

# sets the name of the component to start
runComponent = package + '/' + activity

# sets the name of the component to start
runComponent = package + '/' + activity

# Runs the component
device.startActivity(component=runComponent)
```

```
# Wait for few seconds
MonkeyRunner.sleep(10)

# Create instance of Easy device
easyDevice = EasyMonkeyDevice(device)


# Presses the Start button on landing page
easyDevice.touch(By.id('id/startButton'), 'downAndUp')

# Wait for few seconds
MonkeyRunner.sleep(5)

# Takes a screenshot
result = device.takeSnapshot()

# Writes the screenshot to a file
result.writeToFile('activityTwo.png','png')

# Wait for few seconds
MonkeyRunner.sleep(2)

#Load refernceshapshot from Disk
refImg = MonkeyRunner.loadImageFromFile(path='reference.png')

#Compare two snapshots. for 99% similarity
ifresult.sameAs (refImg, 99.0) :
print 'PASS
else :
print 'FAIL'
```

Note: Please note that com.android.monkeyrunner.easy class of monkeyrunner used in the code snippet above is still (Android Jellybean) under development and may change without notice.[9]

5.4 Pros

1. MonkeyRunner can be used for testing on an actual device without any need to export screens to the desktop
2. Good documentation
3. MonkeyRunner can take screenshots for offline analysis. Screenshots can also be used for doing automated image comparisons against reference images using an inbuilt functionality or using external tools.
4. Monkeyrunner can be extended using a plug-in architecture
5. Monkeyrunner can be used to control multiple devices at the same time

5.5 Cons

1. Requires some knowledge of the application code; for example the user need to know the name of the activities and views.

[9] Reference <Android Source Base>/sdk/monkeyrunner/src/com/android/monkeyrunner/easy/README

2. MonkeyRunner is a low level API based tool – there is no GUI/IDE that is provided along with it by default.

6.0 UITESTAUTOMATION (FOR IOS)

Instruments is a performance, analysis, and testing tool for dynamically tracing and profiling OS X and iOS code.
Automation is a type of template in Instruments, which allows performing testing of user interfaces on iOS device/simulator. Testers can write test cases in JavaScript utilizing the UI Automation APIs (provided by Apple) and automate them using the Automation template. It enables testers to quickly track regression and performance issues. Automation comes with an in-line editor and it supports recording of events/gestures in an easy manner. These scripts/tests can be used to not only automate a user's interaction with the application, but can also contain “asserts” which can test if the application is behaving according to the developer's expectations.UITestAutomation is a part of the Automation template that Instruments invoke for the purpose of UI test automation.

6.1 Various Instruments tools

Instruments not only provides GUI Test automation tool but it has various other automation tools, which help a developers perform various kind of tests related to memory leaks, graphics, animation, time profiling and other areas as shown below:

6.2 Setup and Installation

Instruments is part of the Xcode developer tool suite and it come as a pre-bundled tool with Xcode 4.0+.

6.3 Automated GUI Testing

UI Automation is an application's User tests to be run against elements or their value Additionally, one can log performance data load.

instrument that uses JavaScript to drive interaction with an Interface. These scripts can be structured as a series of the application. Based on the presence of certain UI (object id), these tests can be made to pass or fail. script repetitive tasks to run over a long duration, and then using other instruments, such as memory usage or CPU

There are however some limitations to the UI Automation instrument. If running against a device, that device must be running iOS 4.0+ and it must have hardware capable of multitasking (this won't work on iPhone 3G or the second generation iPod touch).

6.3.1 Writing GUI Test Scripts

To get started with UI Automation testing, one will need to launch Instruments and select the Automation instrument. One can start with a blank document and based on the application need can write test scripts in simple JavaScript.

Below is a very simple GUI test script targeted for an iPad application. This script will generate random touch events for the application.

```
// #### Random Event Generator for iPad####
var width = 1024;
var height = 768;
varmTarget = UIATarget.localTarget();

// tap randomly on screen
vartapTest = function()
{
varyPos = Math.floor(Math.random()*width +1);
varxPos = Math.floor(Math.random()*height +1);

varimageRect = mTarget.frontMostApp().mainWindow().rect();
if(xPos> (imageRect.origin.x + imageRect.size.width))
xPos = imageRect.size.width/2;
if(yPos> (imageRect.origin.y + imageRect.size.height))
yPos = imageRect.size.height/2;

try{
mTarget.tap({x:xPos,y:yPos});
mTarget.delay(1.0); // waiting for 1 second
    }
catch(error){
UIALogger.logError(error);
mTarget.logElementTree();
UIALogger.logFail("Tapping failed @ x: " + xPos + ", y: "+yPos);
    }
mTarget.logElementTree(); //logs elements of the current Tree {mTarget}
}

tapTest();
```

6.3.2 Interacting with user interface elements

One can simulate user interaction with the application interface in a number of ways. If we have buttons or other touch-responding elements, we can simulate a variety of touches and gestures. First, we need to find the correct element. To do that, one can use manual logging and traversal of the UI elements as described in the code fragment above, or one can do a more generic reference based on element type and name.

For example, the following will tap the button named “Equation library”, which is accessible via the root view in the application:

```
// #### Getting Element ####

rootView.buttons()["Equation library"].tap();
```

What's happening here is that we're accessing an array containing only the buttons within this view, then finding the one with the name Equation library. Once we have that item, we use the tap() function to simulate a user tapping on the button.

6.3.3 Supported Events/Gestures

Instruments supports the following events/gestures:.

- tap()
- tapAndHold(Number duration)
- doubleTap()
- twoFingerTap()
- touchWithOptions(Object options)
- dragInsideWithOptions(Object options)
- flickInsideWithOptions(Object options)
- scrollToVisible()

6.4 Pros and Cons

6.4.1 Pros

- JavaScript and UI Automation JavaScript API are fairly easy to learn.
- One can combine the various templates of Instruments with Automation.For example, start the test script and add Leaks / Allocations / Core Data / Time Profiler / System Trace / etc. to test other aspects of the application (memory usage, threads created, heap allocations, battery usage, and many more) while automating them. No other testing tool provides such facilities integrated together so conveniently.
- More closely linked to device and works with device and simulator.
- Supports all gestures (pinch, zoom, swipe, flick, long press, scroll, etc.)
- Apple community support (apart from forums like stackoverflow).
- Able to run native commands like ImageMagic shell command or other scripts to compare images along with advanced image comparison algorithms.

6.4.2 Cons

- Requires linking with application to produce a special build
- Tester needs to write custom test script to get consolidated report of all test cases .

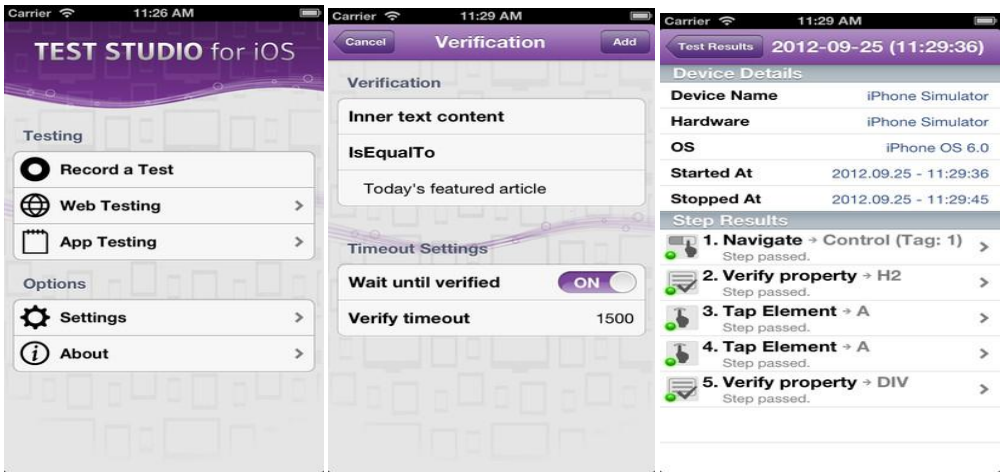
7.0 TEST STUDIO (FOR IOS)

7.1 Introduction

Test Studio by Telerik (http://www.telerik.com/automated-testing-tools) allows developers to record and execute automated tests in mobile applications and websites. Test Studio for iOS is an automated and ad-hoc testing solution for iPhone, iPod Touch, and iPad. It does not require a jail broken device, and it does not use image based element detection. Instead, Test Studio generates queries capable of finding elements based on properties unique to them. In doing so, this allows tests to still be reliable if developers change the location of an element on the screen, or if they change properties unrelated to the initial generated query

7.2 Setup and Installation

Test Studio provides an iOS application to manage application testing. This iOS application is free and can be downloaded from the Apple App store (https://itunes.apple.com/us/app/test-studio/id523796105?mt=8). In order to test a native application, the user needs to download a testing extension from www.telerik.com/ios-testing (it requires a free registration with Telerik)



Test Studio Application UI

7.3 How Test Studio for iOS works

To make a native app Test Studio-compatible, we do need to compile it with a test studio static library (libTestStudioExtension.a) which allows the Test Studio App to interact with the target application on the device. In the testing mode, one can create a series of validation sequences. These sequences consist of interaction events, which one can record directly from the app, and verification tests, which we can specify in the built-in action editor. These tests ensure that user interactions match our specified test criteria.

One can also download a file bundle that contains an Xcode workspace, the Test Studio Extension, the Test Studio app for the iOS Simulator, and a Demo Application[10].

[10] https://www.telerik.com/login.aspx?ReturnUrl=%2fcommunity%2flicense-agreement.aspx%3fpId%3d969

7.4 Pros and Cons

7.4.1 Pros

- Does not require scripting knowledge for test development. It allows test cases to be recorded with required actions specified for each object. Tester can create test cases with just clicks and points. It can test native as well as HTML5 applications.
- Provides a consolidated report for all the executed test cases.
- Detects elements by their properties not location/image

7.4.2 Cons

- Still in beta phase (as of Jan 2013), not very stable.
- Adding other plug-ins (Test Studio's load and performance) is not possible as of now.
- Commercial tool

8.0 FROGLOGIC SQUISH

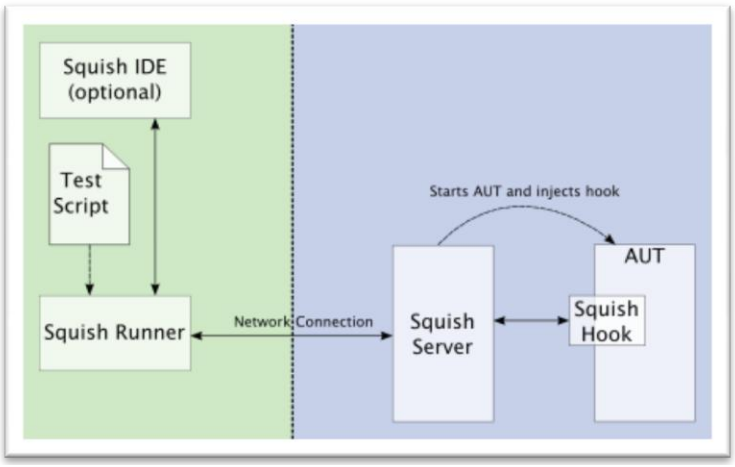
8.1 Introduction

Froglogic provides wide range of cross-platform GUI test automation tools, which specifically support the automation of tests based on QT. Froglogic has three main products:

I. Squish GUI Testing
II. Squish Central
III. Squish COCO

8.2 How Squish Works

Squish runs a small server (squishserver) that handles the communication between the Application Under Test (AUT) and the test script. The “squishserver” starts the AUT and injects the *Squish* hook into it. The hook is a small library that makes the AUT's lives running objects accessible and that can communicate with the squish server. With the hook in place, the squish server can query AUT objects regarding their state and can execute commands—all on behalf of the squish runner. The following diagram illustrates how the individual *Squish* tools work together.



Squish supports QT based mobile application development. It also has support for native iOS application testing.

8.3 iOS native Apps testing

Testing iOS native Apps on an actual iPhone or iPad device is less convenient as compared to simulator testing. To do this one must add a *Squish*-specific wrapper library to Xcode, make a small modification to the application's `main` function as written below:

```
#import<UIKit/UIKit.h>
#import "AppDelegate.h"

#if defined(SQUISH_TESTING) && !TARGET_IPHONE_SIMULATOR ❶
externboolsquish_allowAttaching(unsigned short port); ❷
#endif

int main(intargc, char *argv[]) {
NSAutoreleasePool * pool = [[NSAutoreleasePoolalloc] init];

#if defined(SQUISH_TESTING) && !TARGET_IPHONE_SIMULATOR ❸
squish_allowAttaching(11233);
#endif

intretVal = UIApplicationMain(argc, argv,
nil,NSStringFromClass([AppDelegate class]));
    [pool release];
returnretVal;
}
```

Once the main function is setup correctly for Squish, then the developer needs to add a static library "`libsquishioswrapper.a`" into the Xcode project. This library is shipped with the *Squish* package and can be found in the package's `lib` directory.

8.4 Pros and Cons

8.4.1 Pros

- Provides script less test creation interface to tester similar to TestStudio.
- It supports separation of test data from test cases, which allows reusability of test cases.
- It supports composite logging in XML format and has a log viewer module,
- Supports five scripting languages – JavaScript, Perl, Python, Tcl & Ruby.
- Source code shipped with the tool so that one can enhance the tool for any extra support.

8.4.2 Cons

- Commercial license
- iOS module doesn't support memory leak info, performance or load testing.

9.0 FRANK

9.1 Introduction

Frank is an open source UI testing tool for native iOS apps. Frank uses Cucumber [11]and JSON through which once can write structured test/acceptance tests and have them execute against the iOS application. Frank also includes a powerful “app inspector” (called Symbiote) that one can use to get detailed information on the running application.

9.2 Sample Testscript

- Create a file named navigation.feature inside Frank/features/ and write the test case in the file:

```
Feature: Navigating between screens

Scenario: Moving from the 'Home' screen to the 'Events' screen
Given I launch the app.
Then I should be on the Home screen
When I navigate to "Events"
Then I should be on the Events screen
```

- Create a step definition file called ‘navigation_steps.rb’ inside Frank/features/step_definitions/ write step definitions as follows:

```
Then /^I should be on the Home screen$/ do
pending # express the regex above with the code you wish you had
end
When /^I navigate to "(.*?)"$/ do |arg1|
pending # express the regexp above with the code you wish you had
end
Then /^I should be on the Events screen$/ do
pending # express the regexp above with the code you wish you had
end
```

- Execute cucumber: cucumber features/navigation.feature

9.3 Pros and Cons

9.3.1 Pros

- Comes with Symbiote, a free application inspector, which you can use to get detailed information on your running app.
- Record video of test runs to show the application in action.
- Works for both device and simulator
- Clean CSS-like selector syntax, increases readability.

9.3.2 Cons

- Very limited support for gestures.
- HTML app support is missing.
- As compared to other testing tools, it requires modifications in configuration files to run on real device.

[11] http://cukes.info/

10.0 FEATURE COMPARISION: SUMMARY

Based on the description above, this chapter provides a summary of how the different tools compare

Tool Name	Mobile Platform	Level of Automation	Native App Support	Web Apps Support	Scripting Skills required	Test Recording	Ease of use	Whitebox Testing	Blackbox Testing	License
Instrument		●	●	●	◕	●	◕	●	○	Commercial. Part of X-Code tools
Sikuli		◕	◗	◗	◔	●	◕	○	●	GPL 3
Monkeyrunner		◗	●	●	●	◕	◗	●	●	GPL3
UI Exerciser Monkey		◗	●	●	●	◕	◗	○	●	GPL3
Squish		◗	◗	●	●	○	◔	●	○	Commercial
Test Studio		●	●	●	◔	◗	◕	●	○	Commercial
Frank		◗	◗	○	●	○	◔	●	○	GPL3

● ◕ ◗ ◔ ○

Complete None

11.0 REFERENCES

- http://sikuli.org/
- http://groups.csail.mit.edu/uid/projects/sikuli/sikuli-uist2009.pdf
- http://developer.android.com/tools/index.html
- http://developer.android.com/tools/help/monkeyrunner_concepts.html
- http://developer.apple.com
- http://code.google.com/p/robotframework/
- http://docs.blackberry.com/en/developers/deliverables/11958/Event_injection_808536_11.jsp
- http://www.telerik.com/automated-testing-tools/support/documentation/mobile-testing/deployment/uikit-testing.aspx
- http://www.telerik.com/automated-testing-tools
- http://doc.froglogic.com/squish/4.2/testing..iphone..apps.on.an.iphone.html
- http://www.sunsetlakesoftware.com/sites/default/files/Fall2010CourseNotes/ui%20automation.html
- http://forums.pragprog.com/forums/248/topics/10999

PROPRIETARY NOTICE

APPENDIX A ABOUT HUGHES SYSTIQUE CORPORATION

HUGHES Systique Corporation (HSC), part of the HUGHES group of companies, is a leading Consulting and Software company focused on Communications and Automotive Telematics. HSC is headquartered in Rockville, Maryland USA with its development centre in Gurgaon, India.

SERVICES OFFERED:

Technology Consulting& Architecture: Leverage extensive knowledge and experience of our domain experts to define product requirements, validate technology plans, and provide network level consulting services and deployment of several successful products from conceptualization to market delivery.

Development & Maintenance Services: We can help you design, develop and maintain software for diverse areas in the communication industry. We have a well-defined software development process, comprising of complete SDLC from requirement analysis to the deployment and post production support.

Testing : We have extensive experience in testing methodologies and processes and offer Performance testing (with bench marking metrics), Protocol testing, Conformance testing, Stress testing, White-box and black-box testing, Regression testing and Interoperability testing to our clients

System Integration : As system integrators of choice HSC works with global names to architect, integrate, deploy and manage their suite of OSS, BSS, VAS and IN in wireless (VoIP & IMS), wireline and hybrid networks.: NMS, Service Management & Provisioning .

DOMAIN EXPERTISE:

Terminals

- Terminal Platforms :iPhone, Android, Symbian, Windows CE/Mobile, BREW, Palm OS
- Middleware Experience & Applications :J2ME , IMS Client & OMA PoC,

Access

- Wired Access : PON & DSL, IP-DSLAM,
- Wireless Access :WLAN/WiMAX / LTE, UMTS, 2.5G, 2G ,Satellite Communication

Core Network

- IMS/3GPP ,IPTV , SBC, Interworking , Switching solutions, VoIP

Applications

- Technologies :C, Java/J2ME, C++, Flash/lite, SIP, Presence, Location, AJAX/Mash
- Middleware: GlassFish, BEA, JBOSS, WebSphere, Tomcat, Apache etc.

Management & Back Office:

- Billing &OSS , Knowledge of COTS products , Mediation, CRM
- Network Management :NM Protocols, Java technologies,, Knowledge of COTS NM products, FCAPS, Security & Authentication

Platforms

- Embedded: Design, Development and Porting - RTOS, Device Drivers, Communications / Switching devices, Infrastructure components. Usage and Understanding of Debugging tools.
- FPGA &DSP : Design, System Prototyping. Re-engineering, System Verification, Testing

Automotive Telematics

- In Car unit (ECU) software design with CAN B & CAN C
- Telematics Network Design (CDMA, GSM, GPRS/UMTS)

BENEFITS:

- **Reduced Time to market** :Complement your existing skills, Experience in development-to-deployment in complex communication systems, with key resources available at all times

- **Stretch your R&D dollars :**Best Shore" strategy to outsourcing**,** World class processes, Insulate from resource fluctuations

CONTACT INFORMATION:
phone: +1.301.527.1629
fax: +1.301.527.1690
email: whitepaper@hsc.com
web: www.hsc.com