

Bachelor's Thesis (TUAS)

Exchange Program: European Computer Science

Specialization: Embedded Software

2014

Dominik Kipar

# TEST AUTOMATION FOR MOBILE HYBRID APPLICATIONS

– Using the example of the BILD App for Android  
and iOS



TURUN AMMATTIKORKEAKOULU  
TURKU UNIVERSITY OF APPLIED SCIENCES

Dominik Kipar

# TEST AUTOMATION FOR MOBILE HYBRID APPLICATIONS

This thesis deals with test automation, one of the most challenging problems for software quality and assurance in smartphones and apps. The thesis was commissioned by the QA Department of the BILD GmbH & Co. KG to evaluate test automation solutions and to implement automation scripts for the application BILD App for iOS and Android.

This thesis examines the differences between application types and their special features, advantages and drawbacks as well as how these applications can be tested. The focus of this thesis is on testing for hybrid mobile applications.

One way to achieve high software quality is to optimize testing methods. Software testing can be optimized by establishing test automation. This thesis examines which test cases can and should be automated and for which test cases it is more appropriate to use manual testing.

It is recommended to select an appropriate test automation tool suitable for the environment and that fulfills the requirements for a company and their QA Department. The test company Northway Solution provides 10 general requirements and the company Testlab4Apps published 11 steps to follow in order to select the right test automation tool.

A pre-selection of automation tools, based on the fundamental requirements of BILD, brought five tools for detailed analysis – Calabash, Appium, MonkeyTalk, eggPlant and TouchTest®. After the analysis, the tools and their features were compared to each other. The tool that satisfies the requirements the most is eggPlant with its scripting language SenseTalk®.

Test automation scripts for two critical test cases were implemented. Both scripts are cross platform useable, data-driven and were executed successfully on the BILD App.

According to automation for hybrid applications, eggPlant is a tool that should be considered as a potential solution. The technology for image recognition, provided by eggPlant, is powerful and enables test automation for test cases where other tools fail. Further test scripts need to be created to check if eggPlant is a good solution for native application features as well.

## KEYWORDS:

Test Automation, Test Automation Tool, Mobile Application, Android, iOS, Native Mobile Application, Mobile Hybrid Application, Calabash, Appium, MonkeyTalk, eggPlant, TouchTest

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ  
TURUN AMMATTIKORKEAKOULU

Koulutusohjelman nimi | Suuntautumisvaihtoehdon nimi

Opinnäytetyön | Sivumäärä

Ohjaaja: Tiina Ferm

Dominik Kipar

## OPINNÄYTEN NIMI

(Kirjoita tiivistelmään, maksimi merkkimäärä on 2000).

ASIASANAT:

(Kirjoita asiasanat tähän. Etsi sopivia asiasanoja ONKI -ontologiapalvelun YSA (Yleinen suomalainen aasanasto) ja MUSA (Musiikin asiasanasto) asiasanastoista.

# TABLE OF CONTENTS

## LIST OF ABBREVIATIONS AND ACRONYMS

<b>1</b>	<b>INTRODUCTION</b>	<b>1</b>
<b>2</b>	<b>BILD APP</b>	<b>3</b>
2.1	Mobile Applications in general	3
2.2	Hybrid Application	3
2.2.1	Definition	3
2.2.2	Advantages of Hybrid Applications	4
2.2.3	Disadvantages of Hybrid Applications	4
2.3	About the BILD App	5
2.4	Platforms	5
<b>3</b>	<b>TEST AUTOMATION</b>	<b>6</b>
3.1	Software Test Automation	6
3.1.1	Definition	6
3.1.2	Why tests should be automated	6
3.1.3	When not to automate	6
3.1.4	Test Automation – But what should be automated?	7
3.2	Software Test Automation for mobile Applications	8
3.2.1	Challenges of mobile test automation	8
3.2.2	Benefit of Mobile Test Automation	9
3.3	Software Test Automation for Mobile Hybrid Applications	9
3.3.1	Native Object Recognition	9
3.3.2	Visual Object Recognition (OCR)	10
<b>4</b>	<b>TOOLS FOR TEST AUTOMATION</b>	<b>12</b>
4.1	What is a Test Automation Tool	12
4.2	Requirements for Test Automation Tools	12
4.2.1	No jailbroken or rooted devices	12
4.2.2	True Object Recognition	13
4.2.3	Integration with existing IDE	13
4.2.4	High reusability of scripts	13
4.2.5	Physical device and emulator/simulator support	14
4.2.6	Web application support	14
4.2.7	Data driving, screen capturing and standard reporting capabilities	14
4.2.8	Support for common interruptions and functionality	15
4.2.9	Manual or automatic (scheduled) execution	15
4.2.10	Integration with performance testing tools	15
4.3	Requirements for Test Automation Tool according to the BILD App	16

4.3.1	Already Existing Test Automation	16
4.3.2	Critical Elements for Test Automation for the BILD App	16
4.4	Selection of Test Automation Tools	17
4.4.1	Open Source Test Tool	19
4.4.2	Commercial Test Tool	30
4.5	Direct Comparison of the Tools	36
4.6	Selected Tool	38
<b>5</b>	<b>IMPLEMENTATION OF TEST AUTOMATION</b>	<b>38</b>
<b>6</b>	<b>CONCLUSION</b>	<b>39</b>
	<b>REFERENCES</b>	

## APPENDICES

- Appendix 1. Implementation Test Case 1 “Register a new User”  
Appendix 2. Implementation Test Case 2: Purchase of package 3 with credit card

## PICTURES

- |   |            |
|---|------------|
| Picture 1. Behavior-driven development (Natural Language) in Calabash | 21         |
| Picture 2. Connection list of the eggPlant IDE                        | Appendix 1 |
| Picture 3. Connection list of the eggPlant IDE                        | Appendix 2 |

## TABLES

- |  |    |
|--|----|
| Table 1. Main differences between Visual and Native Object Recognition           | 11 |
| Table 2. Comparison between test automation tools for hybrid mobile applications | 36 |

## LIST OF ABBREVIATIONS AND ACRONYMS

<b>API</b>	Application Programming Interface
<b>App</b>	(Mobile-) Application
<b>AUT</b>	Application Under Test
<b>BDD</b>	Behavior-driven development
<b>CMS</b>	Content-Management-System
<b>CPU</b>	Central Processing Unit
<b>CSS</b>	Cascading Style Sheets
<b>CSV</b>	Comma-Separated Values
<b>DMG</b>	Disk image
<b>ERP</b>	Enterprise-Resource-Planning
<b>EXE</b>	Executable file
<b>GUI</b>	Graphical User Interface
<b>HTML</b>	Hypertext Markup Language
<b>HTTP</b>	Hypertext Transfer Protocol
<b>IDE</b>	Integrated Development Environments
<b>IT</b>	Information Technology
<b>JSON</b>	JavaScript <sup>®</sup> Object Notation
<b>OCR</b>	Optical Character Recognition
<b>OS</b>	Operating System
<b>PC</b>	Personal Computer
<b>QA</b>	Quality Assurance
<b>RAM</b>	Random-Access Memory

<b>ROI</b>	Return of Investment
<b>SDK</b>	Software Development Kit
<b>TXT</b>	Text File
<b>UI</b>	User Interface
<b>VNC</b>	Virtual Network Computing
<b>WYSIWYG</b>	What You See Is What You Get
<b>XLS</b>	Excel Spreadsheet
<b>XML</b>	Extensible Markup Language

# 1 INTRODUCTION

Mobile applications are getting more and more important in our time. The quality of these applications is one of the highest goals for companies, which have developed these applications and provide them to the user. Test engineers in the quality assurance departments of these companies are responsible for ensuring the defined level of software quality.

In 2013 the company BILD GmbH & Co. KG developed a new application – the BILD App – for Android and iOS. These applications provide information in a digital form to the readership of the newspaper BILD. During the development phase the QA department of BILD instructed many testers to verify the quality of the new applications.

The testers executed the test cases manually and the knowledge and experience about testing and applications in general differed from tester to tester. This mixture is good for achieving a high coverage of test cases and to find as many failures as possible.

During this time the first test engineers started with test automation for the hybrid application BILD App. He evaluated different test automation tools and ran test scripts with Appium, Instruments<sup>®</sup> and Calabash. After these test runs and following discussions with other QA teams within the Axel Springer<sup>®</sup> Group, the decision was made in favour of Calabash. Since this evaluation the provider of test automation tools developed new technologies and established new compatibilities. Also new test automation tools, that are worth to be considered, were published to the market – open source as well as commercial solutions.

The Northway Solution Group<sup>®</sup> published a list of requirements that should be kept in mind when evaluating a test automation solution for this project. Additional to the requirements of the Northway Solution Group<sup>®</sup> the software and application testing company Testlab4Apps<sup>®</sup> provides useful information to select the correct test automation tool for a project in the form of an 11-steps-list.



At first a comparison between several test automation tools, whether open source or commercial, was made according to the first two steps of Testlab4Apps® and the requirements of BILD QA: support for the targeted operating systems – iOS and Android – and the used application type – a hybrid application. After this preselecting process a detailed analysis and evaluation of the remaining five automation test tools was accomplished. These test tools are Calabash, Appium, MonkeyTalk, eggPlant and TouchTest®.

The remaining five tools were analysed – following the eleven steps of Testlab4App® and according the requirements of BILD – and compared to each other concerning their details, features, benefits and drawbacks. After this comparison was completed, one test automation tool was selected as potential solution for the test automation for the BILD App – eggPlant.

With the test automation tool eggPlant, two test automation scripts for major critical and important test cases for the software quality and functionality were created and documented. In both cases the created test scripts were able to successfully automate the tests.

Further test automation has to be realised following this work, if the QA Department decides to switch from the current test automation tool – Calabash – to the test automation tool this thesis has evaluated as the potentially best solution for the BILD App.

The QA Department of BILD GmbH & Co. KG commissioned this thesis.

## 2 BILD APP

### 2.1 Mobile Applications in general

A mobile application is a program which is developed to run on mobile devices such as smartphones and tablet computers, and are commonly called app. Normally apps are small in size and individual programs with limited functions like a calculator, a game or simply a mobile web browser. The purpose of those apps is to provide users of mobile devices the same services as on normal PCs. (Janssen 2007)

Mobile Applications can be divided in four categories:

- Native Mobile Application
- Hybrid Application
- Dedicated Web Application
- Web Mobile Application

This thesis focuses on hybrid applications, which are explained following.

### 2.2 Hybrid Application

The main focus of this thesis is on test automation for hybrid applications for mobile devices. They are a special challenge and one of the biggest problems for the QA departments in companies, which are providing hybrid applications.

#### 2.2.1 Definition

A hybrid application combines elements of web application and native app development. By using this technique, developers can combine the familiar technologies and platform independence of Web apps with the power and efficiency of native apps. Advantages of both – web apps and native apps – are combined in one product, without suffering from the drawbacks of each technique. (Buidu 2013 & Vitoriano 2013)

The resulting app still has to be compiled for each platform, because of the different APIs and hardware components of each special device-platform. But the functionality still satisfies the principle of single point of failure.

### 2.2.2 Advantages of Hybrid Applications

Developers with a large knowledge of programming web-based applications (HTML<sup>1</sup>, JavaScript<sup>®</sup>, CSS) can use this knowledge to produce good and powerful mobile applications without the urge to learn a new language.

Native parts provide these apps with powerful and fast extensions, which are beyond the limits of web apps. Further they can be deployed in defined app store of a mobile platform like the Apple<sup>®</sup> App Store<sup>®</sup> or the Google<sup>®</sup> Play Store<sup>®</sup>. This is not possible without the native part of hybrid apps. (Viswanathan n. d.)

The most important advantage is, that the web part is building the common ground for cross-platform applications by using common web APIs.

### 2.2.3 Disadvantages of Hybrid Applications

As mentioned above, it is still required to compile the hybrid application for each target platform. For this the programmer needs the native development tools – SDK – for each platform.

Another drawback is the strong binding to the rules of the different app stores. It is an advantage, but a disadvantage at the same time. From the point of view of web apps, the restrictions of the app stores are much stronger than for Internet browsers, which can lead to problems.

The last big disadvantage is the maintenance – debug, service, package – compared to web apps within an Internet browser. These apps have to be debugged, compiled, packed and uploaded each time to the app stores for changes or fixes.

---

<sup>1</sup> Definition of HTML: <http://www.w3.org/TR/html401/>

### 2.3 About the BILD App

The newspaper BILD is the most read and circulated newspaper in Germany<sup>2</sup> and belongs to one of the biggest publisher in Germany, the Axel Springer Verlag<sup>®3</sup>.

Since the possibility to read newspapers online has become more and more important compared to printed newspapers, Axel Springer Verlag<sup>®</sup> decided to build an online news portal as well. So the website [www.bild.de](http://www.bild.de) was established in 1996. At first there was a website for normal computer and laptops and additionally a mobile view for handhelds and the first smartphones – [m.bild.de](http://m.bild.de) – which provides an easier view of the content with smaller screens of mobile devices.

With the introduction of the Apple<sup>®</sup> iPhone 3G in 2008<sup>4</sup> and the growing numbers of iPhones in use with potential readers, Axel Springer started to develop a special application for the iPhone. This app was available in the Apple<sup>®</sup> App Store<sup>®</sup> in 2009. In the following years BILD developed and published further apps for iPad, Android Smartphones, -Tablets and at last for Windows Phone and Windows Tablets.

### 2.4 Platforms

The Axel Springer Verlag<sup>®</sup> offers the BILD App for following platforms:

iOS 6.0 and newer – Apple<sup>®</sup> iPhone, Apple<sup>®</sup> iPad and Apple<sup>®</sup> iPod touch

Android 4.0.0 and newer – All Android smartphones and tablets

Windows Phone 7.0 and newer – All Windows Phone smartphones

Windows 8.0/Pro/RT and newer – All Windows computer and tablets

All these apps obtain their information via JSON-Feeds from the original CMS-System from which the website [www.bild.de](http://www.bild.de) obtain its information from as well.

---

<sup>2</sup> Statistic about most read German newspapers: <http://www.4imn.com/de/>

<sup>3</sup> Ranking of the ten biggest publisher in Germany: <http://www.mediadb.eu/rankings/deutsche-medienkonzerne-2013.html>

<sup>4</sup> Introduction of the Apple<sup>®</sup> iPhone 3G: <http://www.Apple.com/pr/library/2008/06/09Apple-Introduces-the-New-iPhone-3G.html>

## 3 TEST AUTOMATION

### 3.1 Software Test Automation

#### 3.1.1 Definition

Software Test Automation describes a process, where a computer program is executed against an IT system. This program simulates user interactions to certain software. This can be a database, an ERP-System, an office program. (Ilchenko 2011)

Furthermore, test automation is used for test cases, which cannot be performed by humans or manual interaction with the software. To this category belong API based testing, data-driven testing, model based testing and large-scale result checking for example.

Ilchenko defines software test automation as following, "Software Automation Testing is the process of software verification in which the basic function and test steps, such as running, initialization, execution, analysis and delivery of results are performed automatically by tools for automated testing."

#### 3.1.2 Why tests should be automated

Test automation reduces the time, costs and resources, which are needed to perform a test cycle and decreases the potential of failures at the same time.

Also, test automation can cover an amount of test cases, which cannot be done by human interaction in the same time. Additionally the test coverage is increased, which means that a test engineer can detect failures more efficiently. As a result of test automation, software quality can be verified in a shorter time. (Hayes 1996)

#### 3.1.3 When not to automate

Certain circumstances exist, where test automation should not be used. Some of the examples are explained in the following:

**Unstable design** – This means for example real-time data parts usage within the application. It is not feasible to cover all possibilities of a real-time data with automation, or the test script gets bigger than the application itself.

Another part of this topic is, if the test engineer does not have control over the test environment and data. As a result of this, the investment required to maintain automation here is bigger than the benefit.

Last problem here is a highly configurable application. It is almost impossible to cover all configurations within the test library and attempting to cover these will end up in high maintenance costs and probably will create test failures.

**Inexperienced testers** – Test scripts created by new testers, certainly will not cover and reflect the correct behaviour of an application. Additionally the results of these kinds of testers can confuse developer and test manager. New testers are good manual testers, but the experts should do test automation.

**Temporary testers** – Here it is almost the same as with inexperienced testers. The only difference being, that the testers here are already working in the company and have different professions than software testing. For a fixed time, the temporary testers should test the application from a special view. These persons are good manual testers as well, but not qualified to do the test automation.

**Insufficient time, resources** – Test automation is a long term strategic solution. If the program has to be put on the market in a short time and the company do not have enough time, or well trained testing experts – or testers in general – they should not think about establishing test automation. The needed time to build up the knowledge and finish the test automation will take more time than the company can save by using the tool.

#### 3.1.4 Test Automation – But what should be automated?

Test Automation cannot be used for all kinds of test cases. Costs and workload would be too high compared to the benefit in some cases. So the first thing to do for the QA Department of a company is to determine the test cases, which should be automated. (SmartBear Software® 2011)

Test cases that need large amounts of test data for the same test and that run frequently are appropriate for and should be automated.

The most benefit from automation is granted by following test cases:

Tests that:

- Require multiple data sets
- Run on different platforms (hardware and software)
- Run on different configurations
- Are highly susceptible to human errors
- Repetitive run for multiple builds (for example regression tests)
- Are hard or impossible to run manually
- Require a lot of effort and time to perform manually
- Use high risk condition functionalities
- Show that changes on the system do not cause critical errors (smoke tests)

## 3.2 Software Test Automation for mobile Applications

### 3.2.1 Challenges of mobile test automation

Test automation for mobile applications is hard to realise compared to normal desktop or web applications – due to some additional challenges and problems.

A mobile operating system is normally sandboxing an application. Through sandboxing only a very limited access to internal processes is provided. This makes it more restricted compared to desktop programs.

The general user interface navigation of mobile apps is harder to control. The response time of the interface can vary a lot and is tougher to predict and to work with. Grab and hold interactions are members of this group of interactions for example that can cause trouble.

Mobile devices are in a steady movement. This fact makes a mobile device to be a not statically located entity. That means a current executed test case for an app can break down, if there is a large amount of network interaction or absolutely no network connection at the current location of the device.

Different resolution and size of a screen are a proper reason for a user interface based test to fail, or makes it at least more difficult.

### 3.2.2 Benefit of Mobile Test Automation

Following the most important advantages in form of a catchword list (Hughes Systique Corporation® 2013):

- Testing efficiency
- Improved regressions tests
- Efficiency usage of time (more tests in less time)
- Higher coverage of test cases
- Consistence and repeatability
- Higher resource utilization (24/7 testing)
- Human can perform complex manual test cases

### 3.3 Software Test Automation for Mobile Hybrid Applications

Companies have to handle the requirements of their customers, who are expecting a smooth, clear and high performance application for their expensive smartphones. If the application belongs to a company, which is providing a website of any usage like BILD and their website [www.bild.de](http://www.bild.de), smartphone-owners want an app with higher usability than the original website. High quality and performance of a hybrid app should be the highest goal for companies nowadays. (Kinsbruner 2013)

To reach this goal it is important and necessary to utilize proper testing. The following two methods are used for object recognition in test automation and help the test engineer to create good test scripts:

#### 3.3.1 Native Object Recognition

The first method is native object recognition. This method works with parameters set from the developers in the source code. Kinsbruner describes it as following:

“Object level analysis is used to extract the application object identifier with its properties from the actual native operating system source code, just like the developer used. This is an accurate and fast method to recognize buttons, lists and other objects used by the application.”



### 3.3.2 Visual Object Recognition (OCR)

Visual object recognition utilized the graphical object as images. The parameters from an object are not needed here. Kinsbruner said about OCR:

“Optical Character Recognition essentially uses a smart software engine that converts scanned images of handwritten, typewritten or printed text into machine-encoded text.”

Smartphones have a big problem when it comes to testing, and this makes test automation for hybrid applications complicated and challenging. The variety of providers for smartphones and the different screen sizes and possible web browsers cause trouble for native and visual object analysis.

In order to be able to create test automation for hybrid applications, it is necessary to combine the techniques for native app and web app test automation. Both testing methods cover different aspects of hybrid applications and so it is possible to test and catch any bug or issue in these apps.

In the following table, which shows the main differences between the methods, the pros and cons from the methods also show why it is necessary to use both methods to achieve full test coverage.

<b>Visual Object Recognition</b>		<b>Native Object Recognition</b>	
<u>Pro</u>	<u>Con</u>	<u>Pro</u>	<u>Con</u>
Imperative for end user experience, GUI glitches, implements the WYSIWYG	Slower than Native Object Analysis - requires scan of the screen, become more complex as application screens are much more complex	100% accuracy in native object recognition	Does not detect GUI defects
	Depends on 3rd party software	Much faster than Visual Object Analysis	
	Does not support all languages	Supports all languages	

Table 1: Main differences between Visual and Native Object Recognition

Combining the two methods allows an automation engineer to build a single script, which is portable across devices, browsers and operating systems. Another benefit is a larger scope of test coverage and higher quality for the final product. Further it makes the automation robust and efficient.

Target of this thesis work is to successfully create and apply test automation for the hybrid application BILD App for the iOS and Android platform.

To be able to create successful test automation, the automation engineer has to know which tools are available and which is the best for his problem. Chapter 4 shows information about test automation tools and what should be considered in selecting a tool.

## 4 TOOLS FOR TEST AUTOMATION

### 4.1 What is a Test Automation Tool

With test automation tools an expert is able to create, debug and maintain tests. Those tests can be executed through test automation tools at any time and as often as necessary. He can also analyse results and provide information to the test manager.

Companies started to create test automation tools for other companies. The features and target groups of those tools have increased in the last years. Nowadays companies that start with test automation can choose between lots of different solutions, some open source, others commercial. (Hoffman 2007)

### 4.2 Requirements for Test Automation Tools

This chapter provides an insight into the requirements, recommended by the Northway Solution Group®, which should be guaranteed by a test automation tool. During the process of selecting a tool for a company or project, it is important to keep the requirements in mind. (MacKenzie 2012)

#### 4.2.1 No jailbroken or rooted devices

For better understanding, here is a short definition of the term “jailbroken or rooted device”:

„Jailbreaking or rooting renders the device in a state other than the operating system engineers or device manufacturers intended. The end user, and any application that runs within a jailbroken/rooted device, has escalated privileges to manipulate the system outside of supported methods.“, is said by MacKenzie.

If an application is tested on a device with a modified environment, which differs from the basic settings of the devices from nearly every end-user, the test fails to meet the best practice and this should be unacceptable for a company.

A company should not even consider a testing tool, where it is the only possibility to work with jailbroken or rooted devices.

#### 4.2.2 True Object Recognition

True object recognition is an improvement of bitmap, OCR and coordinate based object mapping and provides the tester with useful and powerful tools for GUI-testing.

According to MacKenzie, “In the example of a web based application, the following UI elements are objects: buttons, text boxes, selection lists, radio groups, images, etc. Every UI object has a set of properties that can be used to identify, define or validate the object.”

By using these properties, it is possible to identify the object, regardless of its position, size and if it is a moving or static object – even it is visible or hidden. As a result of this technique, the automation engineer achieves a high reusable and good maintainable low-cost script development.

True object recognition is the most efficient technique to identify objects and provide access to the properties – given by the developers – and is widely accepted as the best practice according to test automation.

#### 4.2.3 Integration with existing IDE

Developers or test engineers collect in their career knowledge and experience with one or more IDEs over the years. They likely do not want to learn to work with a new IDE “just” for test automation.

In this case it is a huge advantage and highly appreciated, if a test automation tool has the capability to be integrated into the known and already used IDEs. If the test automation tool offers an integration into different – widely spread – IDEs, the opportunity for departments to utilize the test automation is significant higher.

The possibility to use the same tool for all involved developers and test engineers leads to higher efficiency by decreased fragmentation of skillsets and simplifies the entry to test automation.

#### 4.2.4 High reusability of scripts

MacKenzie explains in his article, “Scripts must be reusable across devices and mobile operating system (OS) versions.”

Reusability of scripts is naturally related to true object recognition and should be created to replay interactions on any device running the same OS.

Still it is inevitable to build varying scripts for different operating systems, if the automation engineer is working with native applications or creates scripts, which are able to handle different operating systems with implemented logic for each of them.

Exceptions to this are web based applications, which are working with standard and widely spread browsers. By using true image recognition, a created script should cover all functions and OS.

#### 4.2.5 Physical device and emulator/simulator support

It is not possible for a company to have access to all kinds of physical mobile devices. To reach the highest possible test coverage, it is recommended to apply the 80-20 rule – also known as Pareto principle – by Joseph Moses Juran. (Reh n. d.)

The development and a QA team normally has access to approximately six to eight devices, which should cover the most common devices and OS on the market. Due to this, it is necessary to use emulators and simulators to test the scripts on devices and operating system versions that are not available.

#### 4.2.6 Web application support

As mentioned before, web and hybrid applications have increased in importance over the last years, so it is required that a test automation tool supports these types of applications as well.

„The solution should support all types of applications, native, web based and hybrid, utilizing true object recognition“, said MacKenzie in his article.

#### 4.2.7 Data driving, screen capturing and standard reporting capabilities

For selecting a test automation tool, the three functionalities – data-driven testing, screen capturing and reporting – should be implemented in the tool. Otherwise the automation tool is not appropriate solution. MacKenzie explains it as following:

”Data driving capabilities build on the high reusability requirement by allowing variable data sets and test scenarios to be executed from a single script. Without screen capturing and reporting capabilities (i.e., checkpoints), the solution is extremely limited and is not suited for test automation. These are no-brainers.“

#### 4.2.8 Support for common interruptions and functionality

Possible interruptions such as incoming phone calls and short messages should be handled by the test script, so that the script is not aborted or ends in a fail, but is to be continued correctly after the interruption.

The tool should support functionality and finger gestures like swipe, multi-touch and pinch-zoom.

#### 4.2.9 Manual or automatic (scheduled) execution

It should be possible to start the test scripts manually, if needed, or to organize them by a time schedule, so that the script starts automatically at a preconfigured time.

Good test cases for automatic execution are regression tests.

#### 4.2.10 Integration with performance testing tools

An integration of the automation tool into the existing test environment allows the test engineer to create a homogeneous test framework, which will help to increase the quality of testing and of the application. MacKenzie wrote:

”Research shows that poor application performance translates to lost revenue. The solution should be able to integrate with standard performance testing tools. [...]

In addition, the solution should be able to measure on-device resources such as RAM, CPU, battery and disk space utilization. Add network condition simulation [...] and you have an all-around mobile performance testing solution.”

With this information about the important requirements a test automation tool should satisfy, the following chapters introduce the most important criteria for the automation of BILD App and a selection of open source and commercial tools, according to these criteria.

### 4.3 Requirements for Test Automation Tool according to the BILD App

The chapter 4.2 points out the most important requirements in general for test automation tools. The following chapter points out the requirements for a test automation tool for the BILD App.

#### 4.3.1 Already Existing Test Automation

The QA department of BILD is using Calabash for test automation at the moment. The test engineers have tested Appium for Android and Instruments® for iOS in the past. First results for purchase and login scenarios were achieved with Appium and Calabash during the decision phase. After more test cases and discussions with other companies inside the Axel Springer® Group, the QA department of BILD decided to use Calabash for their automation like it is used for the Die Welt® application. Meanwhile most regression and smoke tests were successfully tested and automated by using Calabash/Cucumber scenarios. Test automation for HTML-elements is making progress with first positive results.

#### 4.3.2 Critical Elements for Test Automation for the BILD App

The BILD App is a hybrid application for common mobile OS, which are using web elements in a native application. Web elements in these applications are used for user authentication and in app purchases. They are connected to different services and servers that are used by the website [www.bild.de](http://www.bild.de) as well. A SAP-system for user authentication and different P4S-servers for each operating system are working in the background. Interfaces developed by BILD build the connection between the applications and the different services.

These embedded web elements are the current problem for the QA department and the test engineers. Additionally several data sets have to be generated and tested to verify that all purchase and user authentication scenarios are covered and can be tested successfully on both operating systems – iOS and Android.

BILD GmbH & Co. KG changed their IT environment a few years ago. A switch from Microsoft® Windows to Apple® Mac OS X was made. It would be a positive feature, if the test automation tool runs on Mac® OS X.

The focus of this thesis is to evaluate and select a test automation tool for cross platform test automation for hybrid applications that is covering these critical test cases for the BILD GmbH & Co. KG and their mobile applications.

#### 4.4 Selection of Test Automation Tools

Nowadays there are a lot different test automation tools available from which a company can choose – open source and commercial tools. The problem is to determine which of those tools offers the most benefits and the least drawbacks for a company or a certain product. This thesis considers the main critical points according to the company BILD GmbH & Co. KG and their product BILD App for Android and iOS.

Both – open source and commercial test tools – are taken into consideration to solve the problems with the test automation for the hybrid applications.

The company Testlab4Apps<sup>®</sup> published eleven steps to select the correct test automation tool for a product or an application. This thesis will follow these steps and connect them with the requirements from the QA department of BILD to distinguish between the different tools and choose the best one for the test automation.

The following eleven steps a company should consider in choosing the proper automation tool (Plotytsia 2014):

##### **Step 1. Which mobile operating systems are supported?**

Check if the test automation tool supports all operating systems on which your application is published and check the earliest and latest supported versions of those operating systems.

##### **Step 2. Which type of mobile application is supported?**

After selecting a set of tools, which can support your operating systems, consider now only those tools, which can support your type of mobile application – native, web or hybrid application. Only a few tools support two types of applications and even less support all three types.



### **Step 3. Is the source code required?**

Since it is not always possible to get the source code – due to security reasons or similar – it is a significant point to consider. The tool should be able to work with the application install file or in best cases with something like app package for iOS devices.

### **Step 4. Is application modification required?**

Some tools require 3<sup>rd</sup> party libraries to be added to the source code of an application to be able to test those. This requires changes within in the source code. It should be considered if this process is wanted.

### **Step 5. In which way are the test scripts created?**

Test scripts can be created through five different methods. They can be created manually or the executed actions can be recorded. Two other approaches are data-driven and keyword-driven testing. The fifth method is called scriptless testing. The test engineer has to decide which method satisfy the requirements and that he wants to use. Afterwards the engineer can check if one or more of the selected test automation tools support his selected scripting method. (Dhall 2008)

### **Step 6. Which programming language is utilized?**

Existing experience in programming languages should be considered when choosing a tool. If the tool language and the application language are well matched, it will be of advantage.

### **Step 7. How the object recognition is done?**

Layout and appearance of an application are changing during the process of development and so they cause problems for automation engineers when it comes to object recognition. Automation engineers have to evaluate how object recognition is realized in a test automation tool, whether it is easy to access all specific objects and if they are easy to handle.

**Step 8. Does the tool support data driven inputs?**

Test automation tools should be able to work and manage data sources. It will enable your script to work with flat files, spreadsheets and database storage to improve test coverage and quality.

**Step 9. How detailed is the result logging?**

Result logs should be able to be customized and provide not only PASS or FAIL information for a certain test case. The log should contain test step, given error message and visual information – such as screenshots – for a failed test case to have more specific information for the developers.

**Step 10. Which options of the integration with other tools are available?**

A test automation tool should be selected in a way, that it fits into the existing testing environment and supports connectivity to the other testing infrastructure components. This can build up strong synergy effects and improve the quality of testing.

**Step 11. What is the pricing?**

For many companies this is one of the most important information. The variation between open source and monthly costs are numerous. If a company considers an open source tool, the staff should check how stable the tool is and how fast it can provide updates for new technologies and OS versions.

For commercial tools the ROI should be calculated and evaluated. If the company cannot afford a certain tool, it should not be considered as a solution. Additional costs can arise, if the tool requires skills that are not available within the QA team.

In the following two subchapters the most common open source and commercial test automation tools for mobile applications are analysed with the requirements from chapter 4.2 in mind and following the eleven steps according to Testlab4App®.

#### 4.4.1 Open Source Test Tool

One point a company should keep in mind by selecting a test automation tool is the provided support for the tool/framework. Some open source tools are supported by a

company, which offers support for payment. If there is no company powering the test automation tool there are online community groups and forums, where an automation engineer can explain his problem and receive help from other automation engineers – often within the next 24 hours. The developers of this software recommend also to use search engines as Google<sup>®</sup> to get further help. (Cindrea 2013)

The following six common and successful open source tools were considered as a solution for the BILD App, but do not satisfy the requirements of BILD digital GmbH & Co. KG:

- Robotium<sup>5</sup> – Supports only Android
- Sikuli<sup>6</sup> – Same basic features but way less powerful then Appium
- Frank<sup>7</sup> – Supports only iOS and Mac
- NativeDriver<sup>8</sup> – Only usable for native applications
- iOS Instruments<sup>9</sup> – Supports only iOS
- Selendroid<sup>10</sup> – Supports only Android

The following three open source test automation tools – Calabash, Appium, MonkeyTalk and – are a pre-selection chosen by their specifications and features. These tools where checked on the bases of the eleven step program of Testlab4App<sup>®</sup>.

This three tools all satisfy the needs for test automation according to the BILD App and the general requirements for test automation from the Northway Solution Group<sup>®</sup>. All of them have different advantages and disadvantages that should be considered. The benefits and the drawbacks are the most important criteria for selecting a tool.

#### 4.4.1.1 Calabash / Cucumber

Calabash is already used by BILD. It is based on Cucumber<sup>11</sup> and can be expressed in so-called natural language (see picture 1). (Xamarin<sup>®</sup> 2012)

<sup>5</sup> <https://code.google.com/p/robotium/>

<sup>6</sup> <http://www.sikuli.org/>

<sup>7</sup> <http://www.testingwithfrank.com/>

<sup>8</sup> <https://code.google.com/p/nativedriver/>

<sup>9</sup>

<https://developer.apple.com/library/mac/documentation/developertools/conceptual/instrumentsuserguide/introduction/Introduction.html>

<sup>10</sup> <http://selendroid.io/>

<sup>11</sup> More Information about Cucumber can be found here: <http://cukes.info/>

```

Feature: Rating a stand
Scenario: Find and rate a stand from the list
  Given I am on the foodstand list
  Then I should see a "rating" button
  And I should not see "Dixie Burger & Gumbo Soup"

  When I touch the "rating" button
  Then I should see "Dixie Burger & Gumbo Soup"

  When I touch "Dixie Burger & Gumbo Soup"
  Then I should see details for "Dixie Burger & Gumbo Soup"

  When I touch the "rate it" button
  Then I should see the rating panel

  When I touch "star5"
  And I touch "rate"
  Then "Dixie Burger & Gumbo Soup" should be rated 5 stars

```

Picture 1. Behavior-driven development (Natural Language) in Calabash

### **Step 1. Which mobile operating systems are supported?**

Calabash supports Android and iOS separately till now. Cross platform support is in the development phase. Other mobile operating systems are not supported and cannot be tested. (Larsen 2012)

### **Step 2. Which type of mobile application is supported?**

Native applications are supported completely by Calabash. By adding and using libraries, it is also possible to create test automation for hybrid applications. JavaScript<sup>®</sup> or Ruby is used and required to identify and interact with web elements inside the application.

### **Step 3. Is the source code required?**

Execution of tests and creating test scripts does not require the source code, neither for Android nor iOS. But Calabash needs modification for the application project, which leads to the next step. (Calabash GitHub n. d.)

### **Step 4. Is application modification required?**

Calabash requires some changes on the source code and the project of the application. Android and iOS requires different changes. For Android it is needed, to install and link the Ruby library and the Cucumber library to the project and apply some changes to the project path settings.

Almost the same changes are required for iOS. The Cucumber and Ruby libraries have to be added here to the project as well and some minor changes to the source code needs to be done.

For both applications it is necessary to compile them with the new information and changes to be able to do test automation with these applications. (Sun 2013)

### **Step 5. In which way are the test scripts created?**

There is no information about capture and playback for Android, but it is realisable with iOS versions smaller then iOS 7.

Capturing is possible by using a real device or a simulator with an application that includes the Calabash libraries within the project. Recording is started by a certain command within the Mac OSX terminal and the recorded steps can be modified later. After modifying the data an automation engineer can play this script back on the device or simulator and it will produce a result log.

Keyword-driven testing is not supported and as a result of this, it is not possible to do scriptless testing as well.

But since Calabash supports Ruby, data-driven testing is possible and should be done. The test scripts have do be created manually inside an editor and has to be compiled and executed for a certain simulator or hardware device. (Belyamani 2012)

### **Step 6. Which programming language is utilized?**

Gherkin<sup>®12</sup> is the utilized language for Calabash and is interpreted by the Cucumber gem. It belongs to the group of behavior-driven development (BDD) languages<sup>13</sup> and is a domain-specific language<sup>14</sup>.

The test steps within the Gherkin<sup>®</sup> scenarios can be programmed in Ruby and embedded into the Gherkin<sup>®</sup> code afterwards.

---

<sup>12</sup> More information about Gherkin here: <https://github.com/cucumber/cucumber/wiki/Gherkin>

<sup>13</sup> Definition of behavior-driven development: <http://www.codeproject.com/Articles/148043/Say-Hello-To-Behavior-Driven-Development-BDD-Part>

<sup>14</sup> Definition of domain-specific language: <http://philcalcado.com/research-on-dsls/domain-specific-languages-dsls/internal-dsls/>

### **Step 7. How the object recognition is done?**

Calabash does not use direct object recognition. To inspect a web view an automation engineer has to use querying and acting on the website via JavaScript<sup>®</sup>. Additionally the element must be visible on the screen to interact with it.

Example for JavaScript<sup>®</sup>:

```
query("webView css: 'a' ").first  
touch("webView css: 'a' ")
```

By using the query-command an object can be identified and the touch-command execute a touch gesture on this button in this case.

### **Step 8. Does the tool support data driven inputs?**

As mentioned in step 5, Calabash is able to create data-driven testing, because it implements Ruby<sup>15</sup> into your project. In Ruby the test engineer can use data-driven programming to include huge sets of data into your program by using list files like CSV or XLS files. (Krukow 2012)

### **Step 9. How detailed is the result logging?**

The logging or report is realized in two different ways. The first way to receive errors or failures is within the output window of the used IDE. Additionally Calabash provides the option to create an HTML document with test case, test step, error message and screenshot from the failure. (Sprindzuikate 2012)

### **Step 10. Which options of the integration with other tools are available?**

There is no information provided on the website or in any GitHub community conversation.

### **Step 11. What is the pricing?**

Calabash is an open source test automation tool and so it is free of charge. It is downloadable from the respective GitHub project.

---

<sup>15</sup> More information about data-driven programming here: <http://www.confreaks.com/videos/1280-rubyconf2012-building-data-driven-products-with-ruby>

Xamarin® Inc. published Calabash and offers trainings for automation engineers. These trainings and further education cost money, depending on the training and level of education.

### **Benefits**

The benefit of Calabash is that it uses Cucumber as library and supports so all types of operating systems and languages. Another important point in this evaluation for a test automation solution is that the Axel Springer® Group already has got some experience with Calabash and is able to create several successfully scenarios already.

### **Drawbacks**

Support for iOS simulator and Android emulator is given, but the functionality of that software is not fully supported by Calabash. Test script should be created with real device to achieve reliable test scripts and results.

#### 4.4.1.2 Appium

The second tool, which was checked for the requirements, is Appium powered by Sauce Labs® Inc.<sup>16</sup>. It is one of the first test automation tools, that are able to provide cross-platform testing, and that is used in several companies. (Appium 2013)

#### **Step 1. Which mobile operating systems are supported?**

Supported operating systems are iOS, Android and FirefoxOS.

#### **Step 2. Which type of mobile application is supported?**

Appium supports all three types of applications – native, web and hybrid applications. Further it is possible to create cross-platform test automation with Appium. This allows automation engineers to create test scripts that run with different operating systems by using the same API.

---

<sup>16</sup> <https://saucelabs.com/home>

### **Step 3. Is the source code required?**

Modifications on the source code is only required for testing real iOS devices. For test automation or running tests on an iOS simulator, Android emulator or real Android device, there are no modifications on the source code needed. Appium follows the rule to test the application, which is uploaded to the application store of an operating system, and not to test a developer edition and an end user edition with varying source code. (Lipps 2013)

### **Step 4. Is application modification required?**

In the case of testing an iOS application on a real device it is necessary to have a valid Apple<sup>®</sup> Developer ID and a configured distribution certificate, which has to be embedded into the source code for the iOS application and compiled with these information. No modifications for Android needed.

### **Step 5. In which way are the test scripts created?**

Test scripts can be created in two different ways. The community of Appium developed a small program with a graphical user interface – the name is appium.app - in which it is possible to capture actions and export these later into every programming language used. Now the automation engineer is able to modify these test scripts and load them back into the appium.app and play them back on a simulator/emulator or hardware device. (SauceLab<sup>®</sup> Inc 2014)

A second option is to write test cases manually using any editor. After finishing the test script, it can be imported into the appium.app and executed from there. Also, scripts can be executed directly from the terminal without the graphical user interface.

### **Step 6. Which programming language is utilized?**

All common programming languages and frameworks are utilized. Appium itself is an HTTP server that can create and handle Selenium<sup>®</sup> WebDriver<sup>17</sup> – former known as Selenium<sup>®</sup> – sessions. Selenium<sup>®</sup> WebDriver is an HTTP API that is implemented in common programming languages and a standard for browser automation. By using this technology it is possible to write the test automation scripts in any language a company

---

<sup>17</sup> More information about Selenium WebDriver: <http://docs.seleniumhq.org/projects/webdriver/>



is already using and has the most experience and knowledge in – for example Java<sup>®</sup>, JavaScript<sup>®</sup>, Python<sup>®</sup>, Ruby, XCode and several more. The other way round, test scripts that are recorded with appium.app can be exported to all of those different languages.

### **Step 7. How the object recognition is done?**

Appium provides true object recognition to the automation engineer. Objects can be recognized and inspected in three different ways for iOS and in four ways in the case of Android:

- By accessibility label
- By element type
- By hierarchy
- By Android ID

These four methods fulfil the requirement for true object recognition and are optimal solutions. For cross platform testing it is necessary that the information for the same object is identical inside the iOS and Android application. Otherwise an automation engineer has to create two different scripts for each platform for the same test case.

### **Step 8. Does the tool support data driven inputs?**

Appium can be understood as a HTTP server which receives test scripts and data from a certain programming language and translates this data to be executed on a target device. Whether data driven or even keyword driven testing is possible, depends on the programming language. If the language supports data driven or keyword driven programming, the method is supported from Appium as well and can be used for test automation.

### **Step 9. How detailed is the result logging?**

Appium only offers result logs in form of debug or error messages in the terminal. If appium.app is used for the execution, the error will appear inside the terminal view of the program. Sauce Labs<sup>®</sup> Inc. offers a possibility to create an account on their website and connect the Appium projects with this account. If the test is executed during a connection to the website is established, it will generate a log file and additional files

inside the Sauce Lab<sup>®</sup> account. The result log contains test data, test steps, screenshots and error messages.

### **Step 10. Which options of the integration with other tools are available?**

Since Appium is using Selenium<sup>®</sup>, it can be integrated into the Selenium<sup>®</sup> testing framework. Companies that are already using Selenium<sup>®</sup> for their web test automation can easily integrate Appium into their framework. Further it is necessary to integrate Appium into the development IDE, which is used by the developers.

### **Step 11. What is the pricing?**

Appium is an open source tool and is absolutely free of charge. Sauce Lab<sup>®</sup> Inc., which is powering Appium, does not offer trainings or courses for this tool. Support and help is offered in form of a GitHub project and several Google<sup>®</sup> Groups.

### **Benefits**

One of the biggest benefits is, that Appium enables the automation engineer to test the same application that the end user is installing on his device. No developer-/testing edition with varying source code is needed here.

### **Drawbacks**

Appium has a big disadvantage: The automation engineer is not able to run a test script on multiple devices simultaneously. That means, that all different devices have to be tested one after another. This costs time and money to the company; especially, if all kind of smart devices have to be tested.

#### **4.4.1.3 MonkeyTalk**

Former known as FoneMonkey, the software MonkeyTalk offers different testing suites for Android and iOS. The provider CloudMonkey LLC<sup>®</sup> has combined the two test suites for Android and iOS to one test suite. The company offers different products. The open source solution MonkeyTalk is analysed in this thesis. The commercial solution MonkeyTalk Pro offers slightly more features, which are not useful for the BILD App. (CloudMonkey<sup>®</sup> 2014)

**Step 1. Which mobile operating systems are supported?**

MonkeyTalk supports both – iOS and Android – operating systems like the other two test automation tools mentioned before and so it fulfils the first requirement.

**Step 2. Which type of mobile application is supported?**

Mobile web applications are not supported by MonkeyTalk till now, but the tool still supports native and hybrid applications. Only if the applications for different operating systems are logically identic, MonkeyTalk will provide cross-platform functionality and will enables the automation engineer to test both applications with the same test script.

**Step 3. Is the source code required?**

Automation engineers do not require the source code to create or execute test scripts, but a modification has to be done to the source code to enable the application to communicate to the MonkeyTalk IDE.

**Step 4. Is application modification required?**

Since MonkeyTalk uses his own IDE – which is based on the Eclipse<sup>®18</sup> IDE – it is necessary that the developers install/embed the MonkeyTalk Agent into the application. This agent is required to enable the communication between the applications – whether the application is running on a real device or simulator/emulator – and the MonkeyTalk IDE.

**Step 5. In which way are the test scripts created?**

Test scripts can be created here in two different ways as well. The automation engineer can use the record feature of the MonkeyTalk IDE or he can create a script by using an editor and save this file with the extension .mt. This test script can be imported into the MonkeyTalk IDE and executed afterwards. The recorded script can be modified within the IDE or exported and modified with any editor and imported to the IDE again for execution.

---

<sup>18</sup> <https://www.eclipse.org/>

## **Step 6. Which programming language is utilized?**

“MonkeyTalk is a functional testing language composed of a simple set of commands for user interface scripting. It’s easy enough to be used by QA engineers and requirements analysts with little or no programming experience, but can also be extended seamlessly with JavaScript for those requiring a bit more programming power. Scripts can be created from scratch as simple text files, or can be created automatically by recording tools such as the MonkeyTalk IDE.”

The MonkeyTalk 1.0 includes a binding to JavaScript<sup>®</sup> already, but the agent is a simple API and can be implemented into any standard programming language like Java<sup>®</sup>, C, C# etc. So with some effort from the developers, MonkeyTalk can be implemented into the existing environment.

## **Step 7. How the object recognition is done?**

MonkeyTalk provides the same object recognition methods like Appium. There are three different methods for iOS and four different methods for Android:

- By accessibility label
- By element type
- By hierarchy
- By Android ID

The test engineer can choose the method he wants to use for object recognition. Like in Appium the object values inside the applications have to be the same for both OS. Otherwise it is not possible to realise cross-platform testing.

## **Step 8. Does the tool support data driven inputs?**

Data-driven testing is possible by using CSV files. The same script can be executed by MonkeyTalk with different values for each iteration. It is even possible to modify the expected results for each execution. This allows the automation engineer to create one script instead of more scripts for the same test case with different expected results.

## **Step 9. How detailed is the result logging?**

The MonkeyTalk IDE provides different types of result logging. The result can be exported as a HTML file with screenshots and step-by-step logging from the test steps.

A XML<sup>19</sup> file can be generated to import it into another logging tool, if used in the QA department.

Furthermore the report can be generated by using the xUnit standard – see step 10.

### **Step 10. Which options of the integration with other tools are available?**

MonkeyTalk can be integrated into other tools, which are using the xUnit<sup>®</sup> standard or into Jenkins. Jenkins<sup>20</sup> is a continuous integration tool, which allows the engineer to manage the test automation as well.

### **Step 11. What is the pricing?**

Pricing is complicated here. The MonkeyTalk IDE is open source and free for download. But no support or help is provided in this “package”. Three different support packages are offered here. Standard support has a yearly fee of 730,00 Euro per year and the premium support cost 2.200,00 Euro per year. Several trainings and courses are offered as well and cost from 7.300,00 to 21.900,00 Euro.

### **Benefits**

MonkeyTalk provides the automation engineer its own IDE with functions for record and playback functionality. Another interesting feature is, that MonkeyTalk itself is keyword-driven. Automation engineers can add new commands to the language. (Intexsoft<sup>®</sup> 2012)

### **Drawbacks**

A general drawback is the lacking support for HTML5 web apps. Another problematic case is an embedded web page inside the application. These are not testable by MonkeyTalk. Test for iOS can only be executed by using Wi-Fi.

## **4.4.2 Commercial Test Tool**

Beside the open source test automation tools, there are many commercial solutions from different provider. Some of them started with functional test automation for

---

<sup>19</sup> Definition of XML: <http://www.w3.org/XML/>

<sup>20</sup> <http://jenkins-ci.org/>

windows application or simple website test automation. Later companies came up with test suites for native and web applications for iOS and Android. In 2012 the first companies started to publish and sell test automation tools for hybrid applications. It is still a new field for automation and the solution strategies of the companies differ in many features and methods. In this thesis two of those commercial solutions are analysed and evaluated according to the requirements of the BILD App and the QA department of BILD.

#### 4.4.2.1 eggPlant by TestPlant®

One of the most common tools for test automation is eggPlant. eggPlant is testing the application under test exclusively by using the user interface of an application like the user sees and uses the application. TestPlant® developed a new technology of non-invasive image recognition which enables the automation engineer to create test cases by using only the graphical interface without knowing or using the source code of those application. This technology makes eggPlant a powerful tool, which is closer analysed below. (Saunders 2013)

##### **Step 1. Which mobile operating systems are supported?**

Supported operating systems in eggPlant are iOS, Android, Windows Phone and Blackberry. Since the requirement only specified support for iOS and Android this requirement is satisfied. (TestPlant® 2014)

##### **Step 2. Which type of mobile application is supported?**

All three types of mobile applications are supported – native, mobile and hybrid apps.

##### **Step 3. Is the source code required?**

No, the source code is not required, because eggPlant is working on the graphical, and therefor the highest layer of an application.

##### **Step 4. Is application modification required?**

For the same reason like in step three, neither a modification at the application nor a jailbreak or root for the device is needed. Only an additional program has to be

installed on the computer and the device to communicate with each other. Here it is a VNC program on both sides of the communicating devices.

### **Step 5. In which way are the test scripts created?**

Two options for creating test scripts are available. First way is to connect your device or simulator/emulator to the IDE and start to capture your actions. eggPlant record the user actions and save the tapped images and the expected actions performed by the tap on the image. In this context image means all kind of objects in the GUI. For an explicit explanation see step seven.

A second possibility to create a script is to do it manually. Test steps and actions are scripted by the automation engineer and these scripts can be executed on a device or simulator/emulator later.

### **Step 6. Which programming language is utilized?**

eggPlant utilized SenseTalk<sup>®21</sup>. This scripting language is a member of the xTalk<sup>®22</sup> scripting languages. It is an English-like language that is easy to read and script. It is not as powerful as other script languages like JavaScript<sup>®</sup> or Python<sup>®</sup>, but it handles the tasks efficiently in the combination with the TestPlant<sup>®</sup> image recognition.

### **Step 7. How the object recognition is done?**

The requirements for true object recognition, which are mentioned before, are not fully satisfied in this automation tool. Image recognition is done on the highest level of graphical interface, so it works with captured and utilized images created by the automation engineer and not with object parameters provided by the source code. As a result of this, the technology is not able to work with hidden objects, but it can handle complex graphical cases, where other automation tools cannot execute the test case successfully. Due to this technology, eggPlant is still considered as a possible solution.

---

<sup>21</sup> More information about SenseTalk<sup>®</sup>: <http://en.wikipedia.org/wiki/SenseTalk>

<sup>22</sup> More information about xTalk: <http://en.wikipedia.org/wiki/XTalk>

### **Step 8. Does the tool support data driven inputs?**

Data-driven testing is possible and realised by using different data types of list files. The automation engineer can use CSV, txt or XML files to provide data sets to a test script and execute these afterwards several times with the given data.

### **Step 9. How detailed is the result logging?**

Results are recorded inside a report suite. The report suite is an extra window within the IDE that provides information about the test script execution. Test case, test steps, execution time, errors and screenshots are provided in this suite. After checking the result within the suite, the report can be exported as HTML or log file.

### **Step 10. Which options of the integration with other tools are available?**

TestPlant<sup>®</sup> offers the integration of eggPlant into following systems:

- Jenkins<sup>®</sup>
- IBM<sup>®</sup> Rational Quality Manager<sup>®</sup> (IBM RQM<sup>®</sup>)<sup>23</sup>
- HP<sup>®</sup> Application Lifecycle Management<sup>®</sup> (HP ALM<sup>®</sup>)<sup>24</sup>

### **Step 11. What is the pricing?**

The price for a single user eggPlant license is 5.520,00 Euro per year and the team license costs 9.200,00 Euro per year. These licenses include creation, managing and execution of test automation scripts. Webinars and information material about new features are included as well. Training and coaching will cause additional costs.

### **Benefits**

Almost every tester can create test scripts; there is no necessarily need for an automation engineer. Testers need good knowledge about the application and the test cases to capture and utilize the correct images and include those into SenseTalk<sup>®</sup> scripts, but they do not need knowledge about the source code or used programming language. SenseTalk<sup>®</sup> is an easy to read, learn and write scripting language, which makes it easy to generate test automation scripts.

<sup>23</sup> More information about IBM RQM<sup>®</sup>: <http://www-03.ibm.com/software/products/de/ratiqualmana>

<sup>24</sup> More information about HP ALM<sup>®</sup>: <http://www8.hp.com/uk/en/software-solutions/application-lifecycle-management.html>



## Drawbacks

Requirements for true object recognition are not satisfied. Some companies are working with hidden elements and control structures to realize test automation for their project. These applications cannot be tested with eggPlant. Another drawback, in contrast to the benefit, can be the utilization of SenseTalk<sup>®</sup>. Most common commands and features for test automation are covered by SenseTalk<sup>®</sup>, but the last update of this scripting language was in 2007. There is a possibility, that features could be developed, which SenseTalk<sup>®</sup> do not cover.

### 4.4.2.2 TouchTest<sup>®</sup> by SOASTA<sup>®</sup>

TouchTest<sup>®</sup> is developed and provided by SOASTA<sup>®</sup>. The company developed two methods for their test automation tool: It is possible to install and use TouchTest<sup>®</sup> on a local machine or within a private cloud solution – running on the secured servers of SOASTA<sup>®</sup>. More information about this is at the end of this chapter. (SOASTA<sup>®</sup> 2013)

#### **Step 1. Which mobile operating systems are supported?**

TouchTest<sup>®</sup> supports only iOS and Android, but it still satisfy our requirement according to supported operating systems.

#### **Step 2. Which type of mobile application is supported?**

TouchTest<sup>®</sup> supports native, web and hybrid applications.

#### **Step 3. Is the source code required?**

The source code is not needed to create or execute test scripts later, but the developer has to make modifications to the application project. (Vila 2013)

#### **Step 4. Is application modification required?**

To enable the application to be tested with TouchTest<sup>®</sup> the developer has to add the TouchTest<sup>®</sup> libraries to the application project. SOASTA<sup>®</sup> provides a prepared tool called MakeAppTouchTestable<sup>®</sup> (MATT) to add the required libraries to the certain application project.

**Step 5. In which way are the test scripts created?**

Scripts can be created here using two different methods as well. Precise gesture recording is possible in TouchTest<sup>®</sup>. The recorded data are translated into so-called App Actions and are added to the embedded clip editor. Manual creation of test scripts is also possible. This is done inside the IDE or using an editor free of choice. Those scripts can be executed with the IDE or combined with captured scripts to generate a more efficient script and achieve higher coverage of test case. (Colin 2013)

**Step 6. Which programming language is utilized?**

Only JavaScript<sup>®</sup> is utilized in TouchTest<sup>®</sup> as scripting language. TouchTest<sup>®</sup> focuses on recording gestures and actions to translate those into App Actions and that those recorded data are played back later repeatedly. JavaScript<sup>®</sup> should be used to specify those recorded test scripts.

**Step 7. How the object recognition is done?**

Object recognition is done on the object layer of the application and satisfies the requirements for true object recognition. That means, that it is possible to identify objects according to their ID or developer set parameters. That presupposes that the automation engineer has knowledge about these parameters.

**Step 8. Does the tool support data driven inputs?**

Data-driven testing is supported by CSV and XML files which can provide the test script with several data sets for test execution. These files have to be implemented into the test script by using JavaScript<sup>®</sup>.

**Step 9. How detailed is the result logging?**

Test results are provided in a certain result view inside the TouchTest<sup>®</sup> IDE. This view contains information about test case, test step, success, error message and screenshots. This result log can be exported as a CSV or XML file. This file provides the same information as the result view, except of the screenshot and visual highlights. (Gardner 2013)

### Step 10. Which options of the integration with other tools are available?
















TouchTest® can be integrated into Jenkins®. With this integration an automation engineer is able to manage the test sets and enable the QA department to use continuous integration for the application. (SOASTA® 2014)

### Step 11. What is the pricing?

The website of SOASTA® is not providing information about the pricing model. On a request for information about annual pricing, the company answers, that information is given after the trial phase and a phone conference first.

#### 4.5 Direct Comparison of the Tools

These five tools – Calabash, Appium, MonkeyTalk, eggPlant and TouchTest® – fulfil the general requirements for test automation tools and provide the automation engineer with different technologies and features. In the following table on the next page the above-mentioned automation tools were compared on the bases of the former mentioned criteria.

Criteria	Calabash 	Appium 	MonkeyTalk 	eggPlant 	TouchTest® 
Support for Android and iOS					
Support for hybrid mobile applications					
Source code required for testing	No	No	No	No	No
Modification of app required	Yes	No	Yes	No	Yes
Test script creation	Recording; manually in editor	Recording; manually in editor	Recording; manually in IDE or editor	Recording; manually in IDE or editor	Recording; manually in IDE or editor
Utilized programming /scripting language	Gherkin® (possibility for additional languages)	All common languages can be utilized	MonkeyTalk, JavaScript®	SenseTalk®	JavaScript®




















Support of true object recognition					
Support of data-driven testing		Depends on utilized language			
Result logging	Output box in IDE; HTML document with full test report	Output box in IDE; Log file on Sauce Lab <sup>®</sup> website possible	HTML document; XML export; xUnit <sup>®</sup> export	Report view in IDE; HTML document	Report view in IDE; XML and CSV export
Integration with other tools	No information provided	Selenium <sup>®</sup> and development IDE	Jenkins <sup>®</sup> and xUnit <sup>®</sup> frameworks	Jenkins <sup>®</sup> , IBM RQM <sup>®</sup> , HP ALM <sup>®</sup>	Jenkins <sup>®</sup>
Pricing Model (annual)	Open Source – Free	Open Source – Free	Open Source – Free	Commercial – From 4.499,00 £ to 7.499,00 £	Commercial – No information
Benefits	Using Cucumber library; Already used by Axel Springer <sup>®</sup>	Tests the same app that the end user is using; Variety of languages	Own IDE as open source tool; Editable Keyword-driven language	Simplicity of SenseTalk <sup>®</sup> ; Cross-Platform technology	Local and Cloud solution
Drawbacks	Limited functionality with simulators/emulators	No simultaneous test script execution	No HTML5 support; not useful for embedded websites	No true object recognition; Development state of SenseTalk <sup>®</sup>	Lack of utilized languages; no price information
Usable/Installable on Mac <sup>®</sup> OS X					
<b>Criteria</b>	Calabash 	Appium 	MonkeyTalk 	eggPlant 	TouchTest <sup>®</sup> 

Table 2. Comparison between test automation tools for hybrid mobile applications

## 4.6 Selected Tool

Based on the analyses accomplished in the chapter before, exemplary test scripts were created and executed with the test automation tools against the BILD App to distinguish, which tool is most useful and suitable for this thesis.

eggPlant enforced itself against the other test automation tools despite the fact that it is not supporting true object recognition and the development state of SenseTalk®.

Reasons for the decision were the attributes and features eggPlant provides to the automation engineer to generate and manage test scripts very easily by using the image recognition technology from TestPlant®. In the case of the BILD App, which is a visual orientated application, this technology is appropriate for test automation. Creating the first functional test script took 10 minutes and it can be used for cross platform hybrid applications.

Chapter 5 is showing how test scripts are created and how two major test cases for the BILD App were solved by using eggPlant.

# 5 IMPLEMENTATION OF TEST AUTOMATION

The first thing to start with, is to install the test automation suite from eggPlant. The installation file is provided from TestPlant® after signing up for the free trial or purchasing the software. Windows users will receive an EXE file and Mac users a DMG file. Both files include an installation wizard for a simple set up of the tool. During the first start of the test automation tool, the user is asked to select a workspace folder on the computer, where the projects are saved by default.

After that it is possible to create the first test script of the first test case. The application is only available in German, so identifiers and strings are only provided in German. The following two test cases are used for testing purchase and authentication test cases. Both test cases require test scripts for web elements inside the application. They show that it is possible to create test automation for hybrid applications and cross platform with eggPlant.

The first test case is for user authentication. This shows the interaction with the web element that is connected to the SAP system – used for user authentication – and that this type of test case is able to automate. The test case and the implementation in the test automation tool are attached as appendix 1: Implementation Test Case 1 “Register a new User”.

The second test case shows interaction with the SAP system and additionally the responsible framework/server for different purchase scenarios. This shows that eggPlant can handle this elements and connections as well. At this test case it is important to include delays into the test script. Otherwise – due to network or server delay – the test case can result in a failure. This test case and implementation are attached as appendix 2: Implementation Test Case 2 “Purchase of package 3 with credit card”.

TestPlants<sup>®</sup> image recognition technology is working well with the BILD App and both test cases could be automated including cross platform feature for iOS and Android and data-driven testing. Most work had to be done by building up the image database for Android – in this case the Samsung Galaxy S4 – and the iPhone by utilizing the captured images from the apps. The implementations are data-driven by using a CSV file and they satisfy the requirements and solve two critical test cases of the BILD QA department without errors.

## **6 CONCLUSION**

The QA department of BILD GmbH & Co. KG selected Calabash for their test automation, because they were expecting synergetic effects with another company of the Axel Springer<sup>®</sup> Group. This company developed the mobile hybrid application Die Welt<sup>®</sup> and successfully applied test automation by using Calabash. They were able to create Cucumber scenarios and test automation scripts to test the BILD App.

The evaluation of the five test automation tools – Calabash, Appium, MonkeyTalk, eggPlant and TouchTest<sup>®</sup> – for hybrid applications with iOS and Android support

shows, that it is worth to consider eggPlant as a solution for the test automation for the BILD App®.

Using the special image recognition and capturing technology of eggPlant in a combination with SenseTalk® enables automation engineers to create test scripts very fast and efficient without requiring strong knowledge about programming language and structure of the application. After setting up an image base from the different operating systems and graphical interfaces it is possible to create cross platform scripts with only small afford.

The two test cases for purchase and authentication scenarios were created within 20 minutes from the scratch and are cross platform and hybrid compatible. Additionally they allow using data-driven testing by providing a CSV file with different data sets.

Calabash fulfils the requirements by BILD as well and can be used to do the test automation for the BILD App, but it is not the most efficient and powerful solution. If the choice is made in favour of an open source tool, the selection should taken on Calabash, otherwise it is reasonable to establish eggPlant as test automation solution.

Further test automation has to be done following this work to verify that eggPlant is able to cover as many test cases as possible according to the test catalogue of the QA department. The result of this thesis covers the test cases for purchases and authentication with embedded web elements inside the BILD App and the connection to the background systems.

## REFERENCES

- Appium, n. a., 2013. *Appium API Reference*. [Online]  
Available at: <http://appium.io/slate/en/master/#appium>  
[Accessed 25.04.2014]
- Belyamani, M., 2012. *iOS Automated Testing With Calabash, Cucumber, and Ruby*. [Online]  
Available at: <http://www.moncefbelyamani.com/ios-automated-testing-with-calabash-cucumber-ruby/>  
[Accessed 23.04.2014]
- Budiu, R., 2013. *Mobile: Native Apps, Web Apps, and Hybrid Apps*. [Online]  
Available at: <http://www.nngroup.com/articles/mobile-native-apps/>  
[Accessed 24.03.2014]
- Larsen, J. M., 2012. *An Overview of Calabash Android*. [Online]  
Available at: <http://blog.lesspainful.com/2012/03/07/Calabash-Android/>  
[Accessed 22.04.2014]
- Calabash GitHub, n. a., n. d.. *calabash-ios Wiki*. [Online]  
Available at: <https://github.com/calabash/calabash-ios/wiki/>  
[Accessed 22.04.2014]
- Cindrea, R., 2013. *Open Source in Mobile Test Automation*. [Online]  
Available at: <http://testausosy.fi/wp-content/uploads/2013/02/Cindrea.pdf>  
[Accessed 23.04.2014]
- CloudMonkey® LLC, n. a., 2014. *MonkeyTalk Coaching*. [Online]  
Available at: <https://www.cloudmonkeymobile.com/monkeytalk/coaching>  
[Accessed 26.04.2014]
- CloudMonkey® LLC, n. a., 2014. *MonkeyTalk FAQ*. [Online]  
Available at: <http://www.cloudmonkeymobile.com/monkeytalk-documentation/monkeytalk-faq>  
[Accessed 26.04.2014]
- CloudMonkey® LLC, n. a., 2014. *MonkeyTalk Language Reference*. [Online]  
Available at: <http://www.cloudmonkeymobile.com/monkeytalk-documentation/monkeytalk-language-reference>  
[Accessed: 26.04.2014]
- Colin, 2013. *TouchTest Execution FAQ*. [Online]  
Available at: <http://cloudlink.soasta.com/t5/Test-Execution-and-Management/TouchTest-Execution-FAQ/td-p/4904>  
[Accessed: 27.04.2014]
- Dhall, S., 2008. *5 Generations of Test Automation Framework*. [Online]  
Available at: <http://qtp.blogspot.fi/2008/10/generations-of-test-automation.html>  
[Accessed: 12.04.2014]
- Gardner, J., 2013. *Export Multiple Results to CSV or XML Spreadsheet Format*. [Online]  
Available at: <http://cloudlink.soasta.com/t5/Knowledge-Base/Export-Multiple-Results-to-CSV-or-XML-Spreadsheet-Format/ba-p/17163>  
[Accessed 27.04.2014]



- Hayes, L. G., 1996, *The Automated Testing Handbook*. [Online]  
Available at:  
<http://www.softwaretestpro.com/itemassets/4772/automatedtestinghandbook.pdf>  
[Accessed 31.03.2014]
- Hoffman, D., 2007. *Software Test Automation: Beyond Regression Testing*. [Online]  
Available at: <http://www.softwarequalitymethods.com/slides/testautobeyondx2-cast07.pdf>  
[Accessed 01.04.2014]
- Hughes Systique Corporation®, n. a., 2013. *Test Automation Tools for Mobile Applications: A brief survey*. [Online]  
Available at:  
[http://hsc.com/Portals/0/Uploads/Articles/hsc\\_whitepaper\\_mobileTestAutomation\\_22Feb2013634971268468845610.pdf](http://hsc.com/Portals/0/Uploads/Articles/hsc_whitepaper_mobileTestAutomation_22Feb2013634971268468845610.pdf)  
[Accessed 01.04.2014]
- Intexsoft® Ltd., 2012. *MONKEYTALK*. [Online]  
Available at: <http://www.intexsoft.com/blog/item/132-monkeytalk.html>  
[Accessed 26.04.2014]
- Ilichenko, A., 2011. *The basic definition of automated software testing*. [Online]  
Available at: <http://blog.qatestlab.com/2011/05/05/the-basic-definitions-of-automated-software-testing/>  
[Accessed 01.04.2014]
- Janssen, C., 2007. *Mobile Application (Mobile App)*. [Online]  
Available at: <http://www.techopedia.com/definition/2953/mobile-application-mobile-app>  
[Accessed 24.03.2014]
- Kinsbruner, E., 2013. *Effective Mobile Test Automation: A Hybrid Approach*. [Online]  
Available at: <http://www.itbriefcase.net/effective-mobile-test-automation-a-hybrid-approach>  
[Accessed 20.04.2014]
- Krukow, K., 2012. *Is it possible to do a data driven testing using calabash-ios?*. [Online]  
Available at: <https://groups.google.com/forum/#!topic/calabash-ios/PtcE0QJeVVA>  
[Accessed 22.04.2014]
- Lipps, J., 2013. *Appium: Mobile Automation Made Awesome - SmartDevCon 2013*. [Online]  
Available at: <http://shelf3d.com/XonoQw6HVPA#>  
[Accessed 25.04.2014]
- MacKenzie, B., 2012. *Top 10 Mobile Application Testing Automation Tool Requirements*. [Online]  
Available at: <http://northwaysolutions.com/blog/top-10-mobile-application-testing-automation-tool-requirements/>  
[Accessed 07.04.2014]
- Plotytsia, S., 2014. *How to Choose the Right Mobile Test Automation Tool?*. [Online]  
Available at: <http://www.testlab4apps.com/how-to-choose-the-right-mobile-test-automation-tool/>  
[Accessed 10.04.2014]
- Reh, J. F., n. d.. *Pareto's Principle – The 80-20 Rule*. [Online]  
Available at: <http://management.about.com/cs/generalmanagement/a/Pareto081202.html>  
[Accessed 10.04.2014]

- Sauce Lab<sup>®</sup> Inc., n. a., 2014. *Appium for Android on Sauce Labs*. [Online]  
Available at: <https://saucelabs.com/appium/tutorial/3>  
[Accessed 25.04.2014]
- Sauce Lab<sup>®</sup> Inc., n. a., 2014. *Selenium Mobile Testing by Sauce Lab*. [Online]  
Available at: <http://saucelabs.com/mobile/selenium-mobile>  
[Accessed 25.04.2014]
- Saunders, N., 2013. *eggPlant for Cross Platform Test Automation*. [Online]  
Available at: <http://assets-production.govstore.service.gov.uk/Giii%20Attachments/TestPlant%20Ltd/Bids/eggPlant%20for%20cross%20platform%20test%20automation.pdf>  
[Accessed 27.04.2014]
- SmartBear Software<sup>®</sup>, n. a., 2011. *6 Tips to Get Started with Automated Testing*. [Online]  
Available at:  
[http://smartbear.com/SmartBear/media/pdfs/6\\_Tips\\_for\\_Automated\\_Test.pdf](http://smartbear.com/SmartBear/media/pdfs/6_Tips_for_Automated_Test.pdf)  
[Accessed 21.04.2014]
- SOASTA<sup>®</sup> Inc., n. a., 2013. *CloudTest<sup>®</sup> Fast, Scalable, Accurate and Affordable*. [Online]  
Available at: <http://www.soasta.com/wp/wp-content/uploads/2012/10/DS-CloudTest-100812.pdf>  
[Accessed 27.04.2014]
- SOASTA<sup>®</sup> Inc., n. a., 2014. *Jenkins CI for iOS Tutorial*. [Online]  
Available at:  
[http://cdn.soasta.com/productresource/download/SOASTA\\_TouchTest\\_Jenkins\\_Tutorial.pdf](http://cdn.soasta.com/productresource/download/SOASTA_TouchTest_Jenkins_Tutorial.pdf)  
[Accessed 27.04.2014]
- SOASTA<sup>®</sup> Inc., n. a., 2013. *Touch and then Test. Precisely*. [Online]  
Available at: [http://cdn.soasta.com/wp/wp-content/uploads/2013/02/SOASTA\\_TouchTest\\_DS-v91.pdf](http://cdn.soasta.com/wp/wp-content/uploads/2013/02/SOASTA_TouchTest_DS-v91.pdf)  
[Accessed 27.04.2014]
- Sprindzuikate, M., 2012. *Calabash Reporting*. [Online]  
Available at: <https://groups.google.com/forum/#!topic/calabash-ios/IWB7Je3zOaM>  
[Accessed 23.04.2014]
- Sun, L., 2013. *Do we need App's Source code for Calabash-IOS*. [Online]  
Available at: <https://groups.google.com/forum/#!topic/calabash-ios/x5sBpNS26Do>  
[Accessed 22.04.2014]
- TestPlant<sup>®</sup>, n. a., 2014. *Data-driven testing*. [Online]  
Available at: <http://www.testplant.com/eggplant/testing-use-cases/data-driven-testing-2/>  
[Accessed 27.04.2014]
- TestPlant<sup>®</sup>, n. a., 2014. *eggPlant Functional*. [Online]  
Available at: <http://www.testplant.com/eggplant/testing-tools/eggplant-developer/>  
[Accessed 27.04.2014]
- TestPlant<sup>®</sup>, n. a., 2014. *eggPlant Integration*. [Online]  
Available at: <http://www.testplant.com/eggplant/testing-tools/eggplant-integration/>  
[Accessed 27.04.2014]
- TestPlant<sup>®</sup>, n. a., 2014. *Read Test Results*. [Online]  
Available at: <http://docs.testplant.com/?q=content/read-test-results>  
[Accessed 27.04.2014]

Vila, R., 2013. *TouchTest Installation and Setup FAQ – Update Sep 2013*. [Online]  
Available at: <http://cloudlink.soasta.com/t5/Mobile-Test-Creation/TouchTest-Installation-and-Setup-FAQ-Updated-Sep-2013/td-p/2247>  
[Accessed 27.04.2014]

Viswanathan, P., n. d.. *The Pros and Cons of Native Apps and Mobile Web Apps*.  
[Online]  
Available at: <http://mobiledevices.about.com/od/additionalresources/qt/The-Pros-And-Cons-Of-Native-Apps-And-Mobile-Web-Apps.htm>  
[Accessed 24.03.2014]

Vitoriano, D., 2013. *Definition of Hybrid Applications*. [Online]  
Available at: <http://de.slideshare.net/idmansp/whitepaper-mobiledeveloperguidance>  
[Accessed 29.03.2014]

Xamarin® Inc., n. a., 2012. *Creating Calabash Tests*. [Online]  
Available at: [http://docs.xamarin.com/guides/testcloud/calabash/creating\\_calabash\\_tests/](http://docs.xamarin.com/guides/testcloud/calabash/creating_calabash_tests/)  
[Accessed 22.04.2014]

Xamarin® Inc., n. a., 2012. *Introduction to Calabash*. [Online]  
Available at: [http://docs.xamarin.com/guides/testcloud/calabash/intro\\_to\\_calabash/](http://docs.xamarin.com/guides/testcloud/calabash/intro_to_calabash/)  
[Accessed 24.04.2014]

## IMPLEMENTATION TEST CASE 1 "REGISTER A NEW USER"

**Test Case 1: Register a new user**

Pre-Condition: No user is logged in and the new user is unknown to the system.

Post-Condition: The user is added to the system; A trial phase is granted; the new user is logged in to the application.

Test steps:

1. Open the settings inside the application
2. Tap on menu item "Mein Konto & Abo" (My account & subscription)
3. Tap on the button "Login"
4. Tap on the button "Konto anlegen" (Create account)
5. Enter valid data for required test fields
6. Activate the check box for the privacy policy
7. Tap on the button "Weiter" (Next)
8. Check if success screen is displayed
9. Tap on button "Weiter" (Next)
10. Check if page "Mein Konto & Abo" (My account & subscription) is displayed and the registered User is logged in

Acceptance Criteria: Post-conditions are satisfied.

Process of test automation:







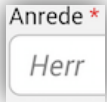
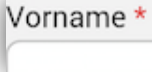
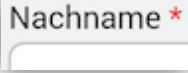
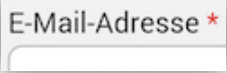
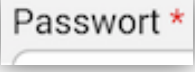
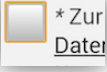

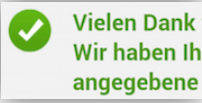


At first the devices, for which the test automation should be created initially, have to be selected. In order to see which devices are connected to the computer eggPlant offers a connection list that shows the devices:

Status	Name	Host	Port	Color	SSH Host	SSH User
●	Windows 7	192.168.42.128	5900	Default		
●	GalaxyS4wifi	10.1.11.172	5901	Default		
●	iPhone	10.1.11.152	5900	Default		
●	GalaxyS4	SCH-I545 (5ac8215900)		Default		

Picture 2: Connection list of the eggPlant IDE

Now the images from the application are captured and utilized, so that they can be used in the test script.

Captured Images and Identifier:

Settings:		Home:	
MyAccount:		NoLogin:	
CreateAccount:		RegiEmpty:	
Address:		FirstName:	
SurName:		Email:	
Password:		NoAGB:	
Next:		Success:	
LoggedIn:		Logout:	
Login:			

Test Script with short explanation what the different commands execute:

```

Put ("GalaxyS4wifi","iPhone") into Phones //Creates List "Phones" with Devices
Repeat with each Phone of Phones //Loop for all devices in the list "Phones"
  Repeat with lines global UserData of file global Path //Loop for all data sets in
  CSV
    Connect Phone //Establish connection to device
    Tap "Settings" //Tap on image "Settings"
    Tap "MyAccount"
    WaitFor 5, "Login" //Delay of 5 seconds, continue if image
    "Login" is visible, else error
    Tap foundimagelocation() //Tap on image that was found in the
    statement before
    WaitFor 5, "CreateAccount"
    Tap foundimagelocation()
    WaitFor 5, "Address"
    Tap foundimagelocation()
    TypeText item global address of ud //Enter data from CSV file "ud" in column
    address
    Tap "FirstName"
    TypeText item global firstname of ud
    Tap "SurName"
    TypeText item global surname of ud
    Tap "Email"
    TypeText item global email of ud
    Tap "Password"
    TypeText item global password of ud
    If imagefound(10,"Success") then LogSuccess "Registration was successful
    //If image "Success" is visible print message
    Tap "Next"
    If imagefound(10,"LoggedIn") then LogSuccess "TestCase was successful!"
    Tap "Logout"
    TypeText BackButton //Press hardware back button
  end repeat
end repeat

```

## IMPLEMENTATION TEST CASE 2 "PURCHASE OF PACKAGE 3 WITH CREDIT CARD"

**Test case 2:** Purchase of package 3 with credit card

Pre-Condition: A user is logged in; the user does not own package 3

Post-Condition: The user is logged in; the user can consume content; the "Offerpage" shows only the package for Bundesliga

Test steps:

1. Open the settings inside the application
2. Tap on "Produkt wählen" (Choose product)
3. Scroll to package 3 "BILDplus Premium" and tap on it
4. Select "Zahlung mit Kreditkarte, Lastschrift oder PayPal" (Pay with credit card, debit or PayPal) as payment method
5. Tap on button "Jetzt Kaufen" (Buy now)
6. Select radio button "Kreditkarte" (Credit card)
7. Enter valid credit card information
8. Tap on button "Weiter" (Next)
9. Check if credit card information are correct and acknowledge with tap on "Jetzt Kaufen" (Buy now)
10. Check if success toast is displayed and tap on button "Weiter" (Next)
11. Tap again on "Produkt wählen" (Choose product) and check if package 1 – 3 disappeared

Acceptance Criteria: Post-conditions are satisfied.

Process of test automation:

At first the devices, for which the test automation should be created initially, have to be selected. In order to see which devices are connected to the computer eggPlant offers a connection list that shows the devices:

Status	Name	Host	Port	Color	SSH Host	SSH User
●	Windows 7	192.168.42.128	5900	Default		
●	GalaxyS4wifi	10.1.11.172	5901	Default		
●	iPhone	10.1.11.152	5900	Default		
●	GalaxyS4	SCH-I545 (5ac8215900)		Default		

Picture 3: Connection list of the eggPlant IDE

Now the images from the application are captured and utilized, so that they can be used in the test script.

### Captured Images and Identifier:

Images from test case 1 in appendix 1 can be reused and do not have to be captured for this test case again. Below are the images that had to be captured for this test case:





Test Script with short explanation what the different commands execute:

```

Put ("GalaxyS4wifi","iPhone") into Phones //Creates List "Phones" with Devices
Repeat with each Phone of Phones //Loop for all devices in the list "Phones"
  Repeat with lines global PaymentData of file global Path //Loop for all data
  sets in CSV
    Connect Phone //Establish connection to device
    Einloggen //Running existing test script for log in
    Tap "Settings" //Tap on image "Settings"
    Tap "MyAccount"
    WaitFor 5, "LoggedIn" //Delay of 5 seconds, continue if image
    "Login" is visible, else error

    Tap "ChooseProduct"
    WaitFor 5, "Product"
    Tap foundimagelocation() //Tap on image that was found in the
    statement before

    WaitFor 5, "SwitchCC"
    Tap foundimagelocation()
    Tap "BuyNow"
    WaitFor 5, "Creditcard"
    Tap foundimagelocation()
    Tap "OwnerCC"
    TypeText item global owner of pd //Enter data from csv file "pd" in column
    owner

    Tap "TypeCC"
    Tap "MasterCard"
    Tap "ValidMonth"
    Tap "Month"
    Tap "ValidYear"
    Tap "Year"
    Tap "NumberCC"
    TypeText item global number of pd
    Tap "TestNumber"
    TypeText item global testnumber of pd
    Tap "Next"

```

```
WaitFor 5, "BuyNow"  
Tap foundimagelocation()  
If imagefound(10, "BuySuccess") then LogSuccess "Purchase was successful!"  
    //If image "BuySuccess" is visible print  
    message  
  
Tap "OK"  
Tap "ChooseProduct"  
If imagefound(10, "Product") then LogError "TestCase was not successful!"  
Else LogSuccess "TestCase was successful!"  
TypeText BackButton //Press hardware back button  
Tap "Logout"  
TypeText BackButton  
end repeat  
end repeat
```