WWW.JAMK.FI

AUTOMATED TESTING OF MOBILE DEVICES

Marek Rosa

Bachelor's Thesis
December 2009

Degree Programme in Information Technology
School of Technology / ICT

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

Author
ROSA, Marek

Type of publication
Bachelor´s Thesis

Date
11.12.2009

Pages
43

Language
English

Confidential
( ) Until

Permission for web publication
(X)

Title
AUTOMATED TESTING OF MOBILE DEVICES

Degree Programme
Degree Programme in Information Technology

Tutor
SALMIKANGAS, Esa

Assigned by
DIGIA Plc

Abstract
The aim was to study the automated testing of mobile devices to assess the challenges that have to be faced and to find solutions to those problems as well as to perform an automated testing and to evaluate its effectiveness.

The theoretical part discusses testing in general and the testing types. The importance of quality assurance in every engineering process is highlighted. It is presented how the subsequent stages of the testing process had been done. Each chapter describes the tester's tasks and what needs to be considered when performing them. Typical problems are discussed and some advices on how to avoid those problems are given.

During the study many new automation scripts were created. Those were successfully used to perform an automated reliability testing of many products. Based on the knowledge that was gained, the assigner's instructions on how to perform automated testing were updated. The testing environment was evaluated and the ideas for improvements proposed. The reporting process was improved by implementing several tools that help to handle the logs produced during the testing round.

Keywords
automated, testing, devices, mobile

Miscellaneous

Contents

1 INTRODUCTION

1.1 Commissioner

Digia is a company that provides information and communication technology solutions to the customers all over the world. Its main expertise areas are smart mobile devices and real-time information systems.

Digia was shaped to be what it is right now on 4th March 2005 when SysOpen Plc, founded in 1990, and Digia Inc., founded in 1997, merged. Current name of the company was established on Annual General Meeting on March 11, 2008 when SysOpen Digia Plc was replaced by Digia Plc (www.digia.com)

1.2 Goals of the thesis

The purpose of this thesis was to study the process of designing and implementing new test cases for automated testing of mobile devices. It is assessed what kind of challenges have to be faced. A solution to those problems is presented. Based on the results of this thesis a new commissioner induction document is created. It concerns matters such as how to design and implement new test cases, how to prepare testing environment and how to maintain the process of testing. Current automation environment is evaluated and feedback given to the company.

2 THEORETICAL INTRODUCTION

2.1 Preface

It is the end of the first decade of the XXI century. Thanks to the technological progress we have access to many useful technologies and solutions, which make our lives easier and richer. However, at some point we realized that we cannot fully utilize many of those facilities, because we cannot carry them with us. What we want now is to have access to them at all times, anywhere.

The mobility of devices is a key factor these days. People want to stay “connected”. The industry responds to this need. If we like to listen to the music while we are jogging, for example, we can carry a mobile Music Player with us. We enjoy taking photos – a compact camera fits in any pocket. Virtually any device comes in portable form nowadays and if we want we can take it with us wherever we go.

But what if we want our camera and our music player with us at the same time? What if we want even more? It was not too long until we realized that carrying a dozen of different kinds of devices is not very convenient and sometimes not even feasible. Again what we want now is to have all the features in one single package. An appliance that combines all the elements we need, which is also customizable, so that it can be adjusted to what we need if for at the very moment. Today's mobile devices try to fulfill these requirements. There are many approaches on how to design such a product. Each company has its own vision of how it should be done. Although there may be many features which are keys to the success, there is one goal which always has to be accomplished to actually achieve it. That is the high quality of the final product.

Nowadays mobile devices are becoming increasingly complicated. Providing users with new functionality must go in pair with thorough testing. More features equals more testing and more testing means more resources needed. It not only increases the cost of the whole process, but also makes

project management a more difficult task. Always when there is a big group of people working on something, problems with communication may occur. It may happen that work is doubled because of this. There may be many reasons for that, like different company, different locations, different language, different time zone etc. This all leads to drop of efficiency and that slows down the whole process, which can have serious consequences like loosing a competitive advantage against other companies.

Mobility is what “drives” consumer products industry today, so the whole mobile devices business is very demanding because many companies want their part in it. Testing does not scale linearly, so adding additional human resources does not mean the results can be expected earlier. That is why improving efficiency is so important for the whole process of testing to go smoothly. Increase of the effectiveness can be achieved by utilizing automation tools.

Checking whether some functionality works as expected requires human intelligence, but repeating the test dozens of times does not. That is the place where computers and automation tools come in handy. Properly designed and executed test case can do the same job as a real tester would do. Instead of repeating the same test steps many times the tester can focus on some other, more challenging and interesting activities. That is not the first time when we come across such a problem. “It is unworthy of excellent men to lose hours like slaves in the labour of calculation which could safely be relegated to anyone else if machines were used." Gottfried Leibniz, XVII century (http://www.thocp.net/timeline/1672.htm)

Mobile device is an “always on” type of device. We do not switch it off when we are finished with our work or we are going to sleep. We rarely ever reboot those devices. It means that the system up-time can often be expressed in weeks or even months, providing that appliance is stable enough. For that to happen, every system element has to work perfectly. Otherwise an application which has stability issues or inefficiently manages its resources would lead to the whole system slow down in a short time, eventually even system freeze. Result of that would be an unpleasant user experience and lost of trust in the

company which produced the device. Comparing to stationary devices the mobile devices can depend on very limited resources, especially power supply. Hence it is vital to optimize their operations, so that they would utilize efficiently those scarce resources. This can only be achieved through careful testing of every device component.

Testing is today an important part of every engineering process. There are many reasons for which we would like to sell a product, which meets its intended quality and customer expectations, reaching from financial, practical, through penal to ethical. In the era of the Internet, news travels very fast and you can be certain that your competitors will make sure that all flaws of your newly released product will be properly highlighted. Building a consumer trust in the company is a difficult and time consuming process. On the other hand, laboriously built firm image can be easily and quickly tattered by a low quality product. Certain defects can be a subject of criminal investigation. For example, when a malfunction of the appliance has caused an injury to a user or other people.

2.1 Testing in general

2.1.1 Test method

“A test method is a definitive procedure that produces a test result. (ASTM definition)” (http://en.wikipedia.org/wiki/Test_method)

Depending on what kind of procedure it is, the test result can help to answer different questions. A test may be failed or passed; a test can provide us with the result which has to be confronted with the reference value. Some test results may put the tested object in to a specific category, for instance, whether tested substance is a ferromagnetic, paramagnetic or diamagnetic. Some tests may be performed by merely using human perception while others require specialized equipment. Easy as it may seem, creating a proper test can be a difficult task. There is no easy way to say that some product is of

good quality. If a test is general then it will produce general results. If it is passed – fine, but if it is failed, then all we get is the fact that it does not work. To find out what exactly failed, more detailed tests are needed. Then again, having a whole set of specific tests, each of which is verifying one specific functionality does not guarantee that the device tested with those will work correctly. That is because of the whole spectrum of possible issues that may arise when several functionalities are used together. That is why testing of complex products like mobile devices is a process. It starts with basics and as the maturity of the appliance increases, tests become more and more complex. Finally, very general ones can be used. Thanks to those, the device as a whole can be tested.

2.1.2 Quality Assurance

Quality assurance is defined as “A program for the systematic monitoring and evaluation of the various aspects of a project, service, or facility to ensure that standards of quality are being met”
(http://www.merriam-webster.com/medical/quality%20assurance)

QA (Quality assurance) is an organized and systematic process, which provides certainty of a product's suitability for its intended purpose. The fact that a product is suitable for a certain purpose does not mean it is perfect. No one can absolutely guarantee that there are no flaws or imperfections present in a product. Perfect quality is unachievable. QA's task is to make sure that the tested object is as close to it as possible.

2.1.3 Accuracy of testing

There exist many different ways of testing the same thing. Some procedures are easy to do and do not require a specialized instrument. Those methods usually lack precision. Sometimes that is enough. Other methods are much more precise but may be expensive or/and consume a lot of time and other resources. Which of those is used depends on many factors. Precise methods have to be used when the tested functionality is crucial for the product to be suitable for the purpose it was designed for. The best example here is

probably life support systems. Carbon oxide detector which starts the alarm after concentration of CO gas is already fatal for human beings is useless. Airbag that does not always work can hardly be called a safety system. Another example (closer to the subject of this thesis) is a mobile device which consumes all its battery power within couple hours.

2.1.4 Reliability of a test

”Test reliability refers to the degree to which a test is consistent and stable in measuring what it is intended to measure”
(http://www.cc.ysu.edu/~rlhoover/OPTISM/reliability_validity.html)

A test is reliable if it gives the same result when it is repeated under the same conditions. The person who repeats the test must know what kind of preconditions have to be met and what tools can be used for this particular test. That is why a test should not only define test steps, but also prerequisites and definitions. When reporting the results of the testing, the tester must provide all relevant information regarding the testing process. This may include issues like versions of testing tools used, environment parameters (temperature, humidity, etc.). If there were any deviations from the rules, those should also be reported along with the explanation why such modifications were made.

2.1.5 Importance of standards in testing

It is very important for all the parties that participate in the testing to agree what standards are to be used. All the definitions have to be consistent. No one can have any doubts about anything that is related to procedures used in testing. Otherwise confusion might be introduced, which may lead to inconsistency in the results. If that is the situation, some test would need to be repeated.

2.1.6 Norms, expectations

Today there are many consumers, government organizations that define various norms to be fulfilled by the product before it can be taken into use and sold. However, there are matters which are not being monitored by such organizations and still can disqualify the device from being sold. Poor robustness of the appliance can result in users filling lawsuits against the company that produced it. That is why testing of the device should start in the early stage of the development process. If the results are not satisfying and there is no progress or it is very slow, then it might be wiser to cancel the development than to continue putting resources on something that has no future.

2.2 Black box testing

“Black box testing is a testing that ignores the internal mechanism of a system or component and focuses solely on the outputs generated in response to selected inputs and execution conditions” (Gao, Tsai & Wu 2003, 119)

This kind of testing can be used to check whether the tested application or component meets the requirements. The principle of black box testing is to compare the monitored output with the expected output. Based on the results of such a comparison it is assessed whether the requirements are met or not.

Depending on what kinds of requirements are used, the Black box testing can be divided into: the functional testing and the non-functional testing.

2.2.1 Functional testing

The purpose of this kind of testing is to test the object for its functional requirements. The main task is to answer the question: Does the application or componenet deliver what it promises to deliver?

2.2.2 Non-Functional Testing

This kind of testing is used to answer the question: How well does the application or component deliver what it promises to deliver?

Many different areas of non-functional testing have been named as follows: (http://www.testingstandards.co.uk/definitions.htm)

- Memory Management
  How efficiently does the application utilize the device's memory?
- Performance
  How robust is the application?
- Stress
  How does the application behave when it is pushed to its limits?
- Reliability
  How often do problems with the device occur?
- Security
  Does the application protect user's data correctly?
- Interoperability
  Is the application compatible with existing solutions?
- Usability
  Is the application user friendly?
- Portability
  Is the application easy to move to the other environment?
- Compatibility
  Is the application compatible with the environment?
- Maintainability
  How much work is required from user to sustain correct operation of the application?
- Recovery
  How well does the application recover from unexpected situation like hardware problem, network problems?
- Installability
  Is the application installation process straight forward?

- Configuration
  How does the application support different hardware and software configuration?
- Conversion
  How well does the application support the migration process of the data from the old system?

2.2.3 Failure types

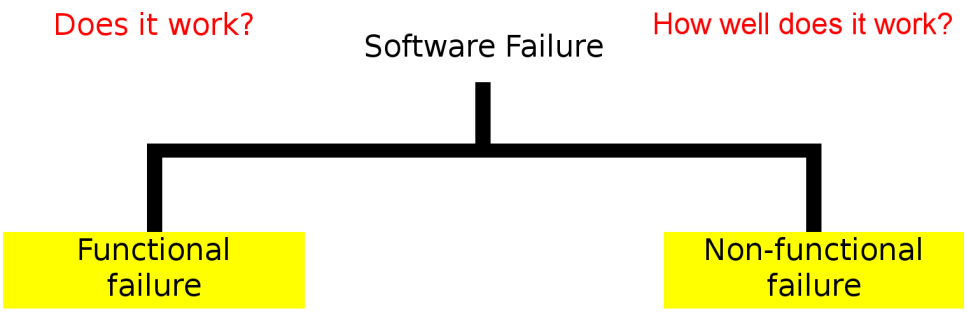

FIGURE 1 Types of failures

As presented on Figure 1 there are two types of software failures: a functional failure and a non-functional failure.

Software failure is a functional failure if it violates functional requirements. It is not possible to perform the operation for which the application was designed for. For example, music playback does not work.

Software failure is a non-functional failure if it violates non-functional requirements. It means that functional requirements may be met, but the application works too slow or consumes too many resources.

2.3 Automated testing

An automated testing is an effective testing solution that can be used to improve the efficiency of testing. One of its main strengths is that it can operate unattended. It allows creating re-usable tests, which can be utilized to perform testing of subsequent builds or software releases. There exist automated tools that can support both functional and non-functional testing. (Watkins 2001, 18)

Creating mobile devices is a long and complex process. A vast amount of human resources is used for the quality assurance. Improving the testing process can not only reduce the amount of resources spent on the development of the product, but also significantly reduces the time needed for the whole process of delivering the appliance to the end user.

An automated testing is very suitable whenever there is a need to perform many repetitions of the same test like in endurance, reliability or regression testing.

By eliminating a human being from the testing process we also eliminate a possibility of a human error. A computer does not make typos and it is never exhausted or falls into a routine. It will repeat the test as many times as we want and it will always obey the rules.

3 TESTING ENVIRONMENT

3.1 Overview

To be able to perform automated testing an appropriate hardware and/or software environment is needed. Not all kinds of testing requires specialized equipment to be used. In those cases a device can be connected directly to a PC running a testing application. It can also be the case that a real device is not available, because of the early stage of its development or for some other reasons.

Even though it is not possible to properly test performance (i.e. power management) without a real appliance, it may still be feasible to perform software testing using an emulator of the device. The emulator is not a perfect solution, because its behavior can vary a bit from the true appliance, but it allows starting a device's software testing earlier than it would be possible without it. It is important because the sooner the error is found the smaller the fixing costs are (See Figure 2) (http://www.chipdesignmag.com/denker/wp-content/uploads/2007/08/cost-of-change.JPG)

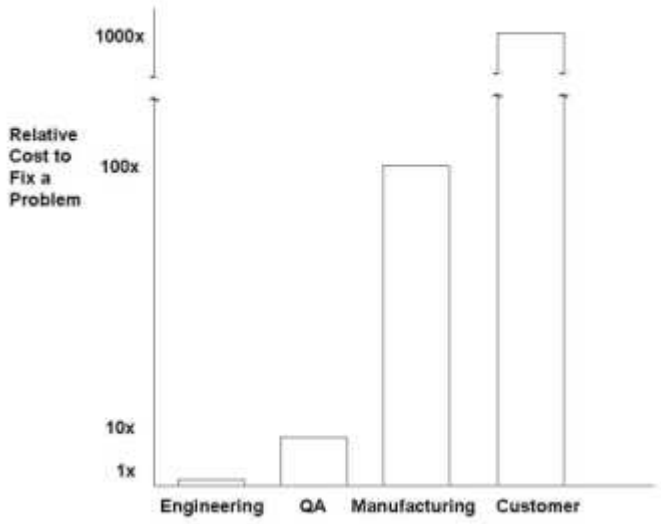

FIGURE 2 Relative cost to fix a problem

3.2 Hardware

The equipment which is used for testing varies depending on what kind of testing is being done. For instance, in power consumption measurement a digital multimeter is used. To increase the reliability of the test and to decrease the measurement uncertainty many iterations of each test are done. An average value is used as an official result. Writing down result of each repetition and then calculating the average value would be unwise – computer can be used for that. In general, nowadays apparatus comes with a computer connection cord and software, which allows operating it using standard PC workstation. Thanks to that it is possible to save the results of the measurement directly to the disk and to process those using appropriate tools.

Working time is only one third of a day. Since computers do not mind working after hours it is rational to utilize them 24 hours a day. Automated testing can be scheduled to run over the night. However, since testing by its nature is being performed on the unfinished devices, which can be defective; it is very likely that one of appliance's flaws may reveal itself during the night testing. Testing tools create logs from the execution, so problems that occur on the device are registered. However, one can never be sure what kinds of problems may arise. During night the test execution is unattended, so it might be more difficult to find out what happened later on. It is always better to have more information on those issues. Video footage of the appliance being tested can often help a lot with understanding on what went wrong. A video camera attached to the testing workstation can be used for that purpose.

3.3 Software

Software used for testing is not consistent among all types of testing. Many of today’s mobile appliances offer a possibility to connect them to other devices, especially computer workstations. There are computer programs designed to cooperate with the appliances connected to the PC. Functionality which is offered this way to the user can be for example: synchronization of the data between PC and the appliance, using the device’s memory as an external

storage for data. This kind of functionality needs to be tested. Hence those publicly available programs are in this case used as a part of software testing environment.

Automation tools used in mobile device software testing consist of three main elements. The first one is the special application running on the appliance. Its main task is to execute commands on the device. This application does not execute test scripts by itself, but it offers an interface, which can be used by other programs to perform operations on the appliance.

The second element is the software installed on computer workstation, which is capable of using interface offered by the first application. It translates script's keywords to a form, which is understandable by first element. The translated command is then sent over to the device using one of possible mediums. Its other task is to determine whether the executed command was successful or not. Depending on what kind of keyword was used different conditions have to be met for the test step to be passed.
This will be covered more closely in the following chapters of this dissertation.

Connection medium, which is used, depends on the appliance. Not all kinds of connection are always supported. However, usually the device can be connected to the computer workstation using a USB cable. Other supported mediums are wireless connection (Bluetooth for instance) and other more specialized types of connections, which may require additional equipment to be used.

The third element of the software environment is also installed on the computer workstation. Usually it is the same machine on which the second element is installed and to which the tested device is attached. The main task of that software is to manage the execution process and to create logs from it. Managing the testing process includes matters like starting, pausing and stopping the execution, determining the order in which specific test cases should be executed and how often.

There are several types of logs which can be taken. For example, a text log containing all the command line messages produced by the second element. Other type is an image based log, which also contains screen captures taken during the execution. There might be also some other kinds of logs, specific to testing area.

Those three elements together create the necessary testing environment. There is a reason why such a division is being done. It would not be wise to design and develop a computer program that can communicate with one specific device. Once the testing is done the program becomes useless or a major rewrite is required. That is why those three layers are introduced. Instead of a specific device a testing tool sees an interface. It knows how to use it and it does not care what is behind it. Any device supporting this interface can be communicated using this testing tool. For the similar reason the execution orders' (test steps) are separated from the testing profile handling. This way creating an overcomplicated tool can be avoided. Smaller elements are easier to manage and can be developed separately by different companies.

3.4 Schematics

Testing environment schematics is presented on Figure 3. The first element of the automation tools is installed on the tested device (nr 1). It communicates with is the rest of the automation tools that are installed on the computer worksation (nr 2). The video footage of the device being tested is taken using a video recording device (nr 3), which saves its logs on the computer (nr 2). If additional equipment (nr 4) is needed to perform the testing then it is also connected to the tested appliance (nr 1) and the PC unit (nr 2).

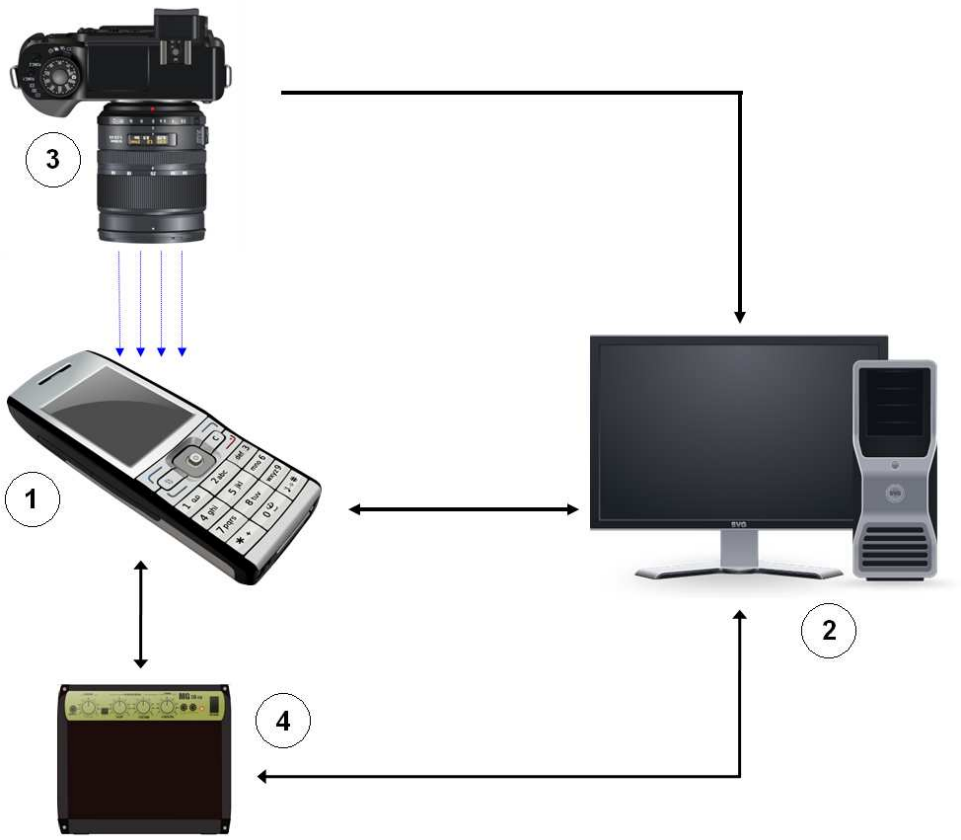

FIGURE 3 Schematics of testing environment

4 TEST SET PREPARATIONS

4.1 Test case designing

A group of test cases that are executed in the same test execution profile are called the test set. Each of the test cases that create test set has to be designed and implemented separately.

A new test case design starts with a customer request. When some new functionality is being introduced then a need for testing it arises. Depending on a specific case the test case specs may be ready or a study is needed. Assuming the former following steps occur. The customer provides test case specifications which explain briefly what the purpose of the test case is, name of the software element which is to be tested and vital steps which have to be followed. Based on those clues the actual test case design is created. Such a design may apply to many different devices, limited number of them or just one. It is like this, because software varies from device to device and thus some elements may be implemented in a different way. Those differences may exist due to the device's hardware designs. For example, the way of entering a text (hardware keyboard, touch UI) Also some extra software elements may be present on the appliance, which enable a new option in certain other applications. Those extra elements may be dictated by matters like user target group, target market, hardware limitations, price and others.

4.1.1 Study

The first thing to be done when designing a new test case for a particular application/functionality is to study the application itself. One must first understand what the application is meant to do, what kind of services it offers. What is the idea behind it, how it should be used and why an end-user might want to use it, under what conditions and in what kind of circumstances.

Thorough study of the application behavior helps to avoid problems in the subsequent phases of test case development. Consequences of negligence

can also manifest itself later, when the test case is actually executed. In such a case it can cause delay in the testing schedule, because the test case has to be corrected. Therefore the results may be available later than originally planned.

Different applications on the appliance may be connected to each other; they may share the same data. This has to be taken into account; otherwise, newly designed test case may interfere with other test cases using the same application/functionality. For example a camera application remembers its last settings, so if one test case modifies those, it will affect other test cases, which use the camera. Applications can also offer their capabilities to user not only directly, but also indirectly by supporting other applications. A File manager application supports Messaging application by allowing a user to start a new MMS by selecting send via MMS from Options Menu. Once a tester has a good understanding of how the application works s/he can decide on what is the most natural way of accessing certain software functionality.

A test case must not assume the current state of the device. It means that first steps in test cases should change the device state to a desired one. This may include issues like appropriate network in use, screen orientation and others. Even though each test case should be designed in such a way, it is still a good idea for every test case to clean up after it is finished. It includes such activities like closing all unnecessary applications, making sure that there are no notes left opened on the device’s screen.

That is, however, not enough to design a good test case. Many applications use data and some of them gather the data. That can have a huge effect on the test case design, because software is usually designed so that it offers only functionality which is relevant in the current context, which makes sense. All the features that do not meet this requirement are hidden. For example, there is no point in displaying menu option edit picture if there are no pictures available or to display scroll bar on the screen when there are only few items available and those fit on the screen perfectly.

Another issue to consider when creating new test case is how a specific application behaves when it is being run for the first time. Often there is some special kind of activity required, some settings, or a configuration has to be entered, updated. For example, in the mail application the mailbox has to be defined. There are two ways of handling such a case. One is to create a list of preconditions which have to be fulfilled in order to make a test case script work properly. That means that the tester who will execute that particular test case later on has to configure the device properly before proceeding with the script execution. That does not mean that setting up the appliance for testing has to be done manually. In most cases a special run-once type of script can be developed. That kind of script is convenient particularly when a lot of input is needed from the user. Not only it can be time consuming, but also raises the chance of a mistake, which can affect the testing later on. The other approach is to make a test case to test application state. If the state is proper then execution may proceed, if not, then some extra test steps have to be performed to achieve it. Out of those two approaches the former one is usually preferred, because it is being run only once, therefore does not create extra overhead to the test case itself. However, it is not always feasible. There also might be other, test case specific factors, which will make the latter approach a better solution.

4.1.2 Designing

Once all those issues are properly studied the actual designing part can start. The main task here is to decide on an order in which operations are being performed and what kind of outcome is expected. The second objective it to determine how and where to verify that the test case is going as planned. Usually there are many ways to verify correctness of the outcome. As there are many ways to do it, there are many matters to consider when deciding on one of those. The first matter is to make sure that the evaluated manner is really false positive proof. The second one is to assess the efficiency of the selected approach. Another one is the portability of a chosen solution. Selecting the wrong way of verifying may render the test case very difficult to adapt to devices other than the one which was used when creating it.

When test case designs are ready, it is time to implement them using appropriate scripting language. It depends on the testing area, what kind of language is used.

4.2 Test case development

The test case implementation is done using appropriate scripting language. The syntax of language used for writing scripts for automated testing is quite simple. It is based on keywords.

4.2.1 Keywords

Each keyword may be followed by parameters, but not all keywords need parameters. For example, taking a screen capture does not require any parameter, but to perform text verification a string, which is to be verified has to be provided. Scripting language offers keywords which can be put in three main groups:

1. Verification keywords
   Those keywords are used to confirm that certain action has been successfully performed or that application is in a certain state. Usually it means that the device's screen is checked for the presence of a specific text or a picture, depending on the context. Pseudo code examples below show typical usage of those kinds of keywords.

```
// Following keyword is used to save a screen capture to the
// execution log.
TAKE_SCREEN_CAPTURE

// This keyword along with a parameter is used to verify
// whether specified text can be found from the device's screen
FIND_TEXT "Hello world"

// This keyword along with a parameter (bitmap name) is used
// to verify that specified image can be found from the screen
FIND_IMAGE "application_1_icon.png"
```

2. Action keywords

   As the name implies, action keywords are used to perform an operation on the device. Those may include matters like pressing a key, tapping certain point on the screen, selecting menu item or starting an application on the device. Some of those keywords are more sophisticated and besides performing an action, they also check that the outcome is correct. For example starting an application in a current state independent manner (if specific application is already started, but it is in an “incorrect” view, then it is closed and then started again. This way “fresh and clean” state can be achieved.

```
// Following keyword along with a parameter is used to imitate
// user pressing the hardware button of the device
PRESS_HARDWARE_BUTTON “Button_1”

// This keyword is used to tap the specified text
// on the device’s screen. Only works on device’s equipped
// with touch screen.
TAP_TEXT “Item 1”

// This keyword along with a parameter is used to select
// one, specified menu item from all available items of the
// menu which is currently visible on the device’s screen
SELECT_MENU_ITEM “Open”

// This keyword is used to start an application specified by
// a parameter
START_APPLICATION “Application_XYZ”
```

3. Diagnostic keywords

   Keywords from this group perform “examination” on a device. Results of such a test can provide much valuable information about the current state of the appliance, like how much battery power is left, memory consumption, time on the device internal clock and others.

```
// Following keyword is used to retrieve the information about
// the device’s memory
CHECK_MEMORY_STATUS
```

```
// This keyword is used to retrieve the information about
// remaining device’s battery power
CHECK_BATTERY_POWER
```

Script is built using keywords from those three groups. Each of those keywords works independently. If one of the test steps is failed then the whole script is failed. All following test steps are skipped and the test iteration is marked as failed.

```
// An example Music Player script
// First step, start a Music Player application
START_APPLICATION “Fast Music Player”
// Second step, open option menu
TAP_TEXT “Options”
// Third step, select ‘Play’ from options menu
SELECT_MENU_ITEM “Play music”
// Fourth step, verify that music is being played
FIND_TEXT “Playing”
FIND_TEXT “Vivaldi – The Four Seasons (winter)”
// Fifth step, stop playing the music
TAP_TEXT “Options”
SELECT_MENU_ITEM “Stop”
// Sixth step, close Music Player application
TAP_TEXT “Close”
```

4.2.2 Localization files

One of the first rules taught on programming courses in any programming language is not to use hard coded values in the program code. Hard coded values are values which cannot be modified without directly editing source code of an application. There are many reasons why such a “technique” or programming “style” is strongly discouraged. For example, if certain hard coded value is used several times, then to modify it one must modify every occurrence of it separately. Programs written like this are difficult to debug,

because hard coded values cannot be traced. Better solution is to use variables, for values, which are changing during the execution or constants for values which are always the same. Idea is to have an identifier for each entity in a program. This way it is easy to modify them or refer to them from other sections of code.

Localization files are used to make scripts portable. If the same script is to be used for many products and each of those products use different language or a different synonyms for certain words (rock, stone, boulder, ore, etc) then one would need to create as many copies of the script as there are products. Also this process would need to be repeated each time a change to script itself is needed. To avoid that, hard coded values must not be used in scripts. Logical names should be used instead.

Revised version of “An example Music Player script” has the following form.

```
// An example Music Player script
// First step, start a Music Player application
START_APPLICATION $music_player_application$
// Second step, open option menu
TAP_TEXT $options_menu_caption$
// Third step, select ‘Play’ from options menu
SELECT_MENU_ITEM $play_music_menu_caption$
// Fourth step, verify that music is being played
FIND_TEXT $playing_music_caption$
FIND_TEXT $test_song_name$
// Fifth step, stop playing the music
TAP_TEXT $options_menu_caption$
SELECT_MENU_ITEM $stop_music_menu_caption$
// Sixth step, close Music Player application
TAP_TEXT $close_application_caption$
```

All the logical names are gathered into a localization file. The structure of this file is table like.

Logical_name	Product 1	Product 2
$music_player_application$	“Fast Music Player”	“Cool Music Player”
$options_menu_caption$	“Options”	“Options”
$play_music_menu_caption$	“Play music”	“Play song”
$playing_music_caption$	“Playing”	“Music is being played”
$test_song_name$	“Vivaldi – The Four Seasons (winter)”	“Chopin – Minute Waltz”
$close_application_caption$	“Close”	“Exit”

Appropriate values for logical names are loaded by testing software depending on what product is to be tested. Thanks to a localization file only one script needs to be maintained.

4.3 Test case verification

Once a particular test case is developed it has to be verified. Depending on the language used by a particular build, a different localization file might need to be used. As the development of the software is progressing, some elements may change. Hence it is not enough to verify testing script at the beginning and then simply use it for testing with each new build. Each test case has to be verified every time new software build is available. Depending on the maturity of the device more or less changes are introduced. Sometimes only localization files tweaks are needed. Another time few test steps require modification and verification may actually turn to re-designing.

Verification is done by running selected test cases several times on the device with desired software build. During those verification executions tester observes the appliance at all times making sure that all the test steps are performed as they should and that the outcome is correct. If there are no problems with the test case then it is marked as verified. It will still be verified at least once before the official testing round starts. It happens during the test set pilot round. Pilot round (Test run) is covered in more details in the next chapter.

5 ENVIRONMENT CONFIGURATION

5.1 Setting up hardware components

Laboratory environment has to be properly configured before any testing can take place. All the necessary components of testing environment are installed and prepared for the specified testing type.

Some kinds of testing may require additional hardware elements to be used during the testing session. For example power management testing requires a multimeter to be properly connected to the device. It might be also necessary to calibrate the measurement equipment. This process does not need to be done each time the apparatus is to be used. Usually, the producer of the equipment provides the information about the interval of necessary periodic inspections. Measurement devices as any other electronic appliances are vulnerable for the environment change. If such equipment is moved to other place, then it should be given some time to acclimatize before it is used or even turned on. One more issue is that declared parameters of such apparatus are only achieved after it has stabilized its operations. It means that each time before it is to be used it requires some time (~30 min) to “warm up”.

Another matter is the video recording device. To be able to capture video footage of the testing process a dedicated video camera is need. Depending on the type of testing, it might be specialized equipment, adequate for the testing purposes or a normal, generally available video camera. As the testing can take several days, there are a lot of “opportunities” for camera to lose the tested appliance from its sight. It might be due to vibration caused by people walking around or any other situations that may happen in a working place. To avoid accidental movements of the camera it is advisable to have a proper support stand for it.

Last but not least is to prepare the tested appliance itself. Mobile device to operate properly may require additional elements to be put inside of it. Those include matters like Battery, Sim card and Memory Card. To make sure that

execution will proceed without any problems caused by any of those elements following issues has to be checked.

A SIM card may happen not to be valid anymore. Its air time may have run out, or it may be running out in a few days, which might be not enough to finalize the testing round. It may also be broken, which means it cannot correctly register itself to the network. Therefore it cannot be used.

Memory card has to be formatted before each new execution. This is needed, because the data that was gathered during previous testing round may otherwise interfere with the data that will be saved to it during current execution.

The consecutive test iterations are usually executed without any pauses. Hence, the battery pack can supply the device with power for only couple hours. A charger is used to prevent the device from running out of power.

5.2 Installing and configuring testing software

The core of the automated testing is automation software. Each element of the environment has to be installed on its respective hardware. The testing tools are in constant development. Therefore the tester should always check that the newest possible versions are used. This is especially important when the new product is going to be tested. It may happen that previous versions of the testing software may not support newer products.

After everything is installed an appropriate configuration settings have to be applied. In the first place the configuration package suitable for the tested product is selected. Secondly the connection settings are adjusted, so that they reflect current conditions.

If there is a need to use some other, possibly third party software for the purpose of the testing round then it also has to be installed and properly configured. The specific installation and configuration instructions depend on the software that is to be used. Those are usually provided with the software

bundle or through an appropriate web page maintained by the software vendor.

Testing logs which are being created during the execution are stored to workstation hard drive. Those testing logs can be quite big, especially video logs. If the execution is long then it may happen that there is not enough free disk space available. To ensure continuity of testing enough disk storage has to be secured before starting the execution.

If there is any additional equipment that needs to be used then it is installed at this point. Instructions on how to do that are apparatus specific and thus are not provided here.

Once all necessary software is installed and configured a connection between PC and the tested device can be initialized. If it is not working then it means that automation software has not been properly configured. It can be either a problem with tools running on the computer workstation, tested device or both. When the connection is working it is time to verify that right product configuration package has been selected and that selected test scripts are correct.

5.3 Preparing testing profile and test run

The test set created by following the instructions from the previous chapter is used to prepare testing profile. The automation tool uses a profile to customize the execution. The execution start time can be adjusted. Individual test cases can be set to run more or less frequently than the rest of test cases in the test set. Video footage recording rules can be changed. It can be decided what kind of logs should be preserved and many other. Thanks to that, the execution round can be tuned to meet the requirements of specific testing area.

A test run is a relatively short testing round performed before actual testing can take place. Even though the connection between PC and device is working it does not mean that automation tools are operating properly. The

purpose of the test run is to verify that all the configuration settings are appropriate and to check for the one last time that there are no problems with testing scripts. Depending on how many test cases are included in the test set the test run can take couple minutes or couple hours.

The test run helps to avoid a situation when official execution has to be stopped because some test scripts have flaws. Stopping the execution is an undesired situation, because it breaks the continuity of the testing. When testing is interrupted and then started again, the logs are in several pieces. It creates additional work to be done when reporting the results of the testing round, because the logs have to be concatenated.

6 ACTUAL TESTING

6.1 Scheduling

Automated testing can be scheduled to start and finish automatically. The tester's task is to make sure that the scripts work correctly and that all the necessary data needed for the testing is available. The testing continues outside normal working hours so it is vital to be sure that everything will work fine while there will be no one to look after the process.

6.2 Execution maintenance

Even though the testing is automated, the execution still has to be maintained. Every day during working hours the tester checks how the execution process performs. There are several reasons why it has to be done. The most important of course is finding software flaws. All the failed iterations are checked to find the cause of the failure. If it is indeed an error of the device then the comments are written to the tester execution notes document and all the necessary logs are copied to some other, tester findings directory. This way it is easier to access those at the later date (no reason to "dig out" through all the logs again). However it might be that there was no problem with the device itself and that the failure was actually causes by problems with automation.

The test cases are checked for any possible interference with other test cases during designing and implementation process. However, it may still happen that something was omitted and the test case misbehaves. It may happen that after the test case has finished its execution it does not restore device's state properly.

Another issue is maintaining the testing environment. In the early stages of the device development it may not be very stable and that means there might be a lot of failed test iteration. By default all the video logs from failed iterations are

preserved. Considering that there might be thousands of iterations already executed, the remaining disk space may be coming short. A situation like this is of course undesired and thus free disk storage has to be monitored. Resolution of such a situation is to either move some of the logs to other storage medium (like other PC unit hard drive, CD or other).

Last but not least is the testing hardware. Situation like this might never happen but it is possible so it has to be considered as well. The PC unit might have a malfunction. Instability of the power supply may cause a system reboot which of course results in immediate interruption of the testing. This is an especially undesired situation if it was to occur during the weekend, when there might be no one to resume the execution.

6.3 Daily reporting

Testing may take several days to complete. Sometimes there might be a serious bug present in the build under testing. In such a situation the end result of the whole device's stability may be heavily biased by a single issue. A customer might find this kind of report not very useful, that is why it is important to provide daily reports on the execution progress and any possible defects found. In some cases such a defect found may result in customer request to stop the testing. There is no point in wasting time on maintaining the execution the results of which are not providing any new information on the stability of an appliance.

Systematic reporting of partial results may also be very helpful for the customer when planning new test execution. It may happen that information provided by the tester in daily report can help customer to decide whether some other testing for that particular software build of the device should be postponed or given a green light.

To provide the status of the testing round the logs gathered by the automation tools are studied. Failed test cases are investigated in order to find out what went wrong and why automation tools were unable to go through all test

steps. Conclusions based on those findings are provided to the customer in a form of a daily report.

6.4 Results gathering

An important element of the automation tools is a log gathering module. Without possibility of saving the logs to the disk storage automated testing would not be useful. Because of the complexity of contemporary mobile appliances the errors that occur on the tested device are usually not very easy to repeat. Hence, it is vital to save as many information about those errors as possible. To fulfill that need three different kinds of logs are taken during the execution. Those are Image based log, Video log and Text log.

Image based log is a primary source of information used to determine how the execution went. From this kind of log it is easy to find the point where the problem manifested itself. Quality of those logs is very dependent on the test script. Verification keywords have to be used quite often, otherwise there might be some black spots in the log.

Video logs might be not easy to use directly, as there is no information about the current test step provided in the log itself. Usually those are used along with Image based logs. First the image log is used to locate the failure point and then video log is to get more information about what kind of problem occurred. Video log size is quite big. Hence, usually only the logs from failed iterations are preserved. Video footage is always recorded, but if the test case turns out to be passed then the log is deleted. In most cases passed iterations videos does not provide any additional information, so this is not affecting the results quality.

Text log contains all the lines produced by the execution tool during the testing round. It may be a little bit difficult to read and is rarely used as a primary source of information. However, it contains more detailed information about the problem from automation tools point of view than other logs. Hence, it is used when the result of the failure cannot be determined by using other

logs or when it is unclear. This kind of log is good for finding problems related to test script or automation tool itself.

All those logs types are complementary to each other. Used together they provide quite good information about the reasons of a failure.

Depending on the area of testing there might be other, different kinds of logs taken. In most cases, however, those are taken independently from the automation tools. Usually separate equipment is used for that purpose and it is connected to the device in a different way than the computer workstation is.

7 REPORTING

7.1 Investigating logs

If the daily reporting has been done properly during the testing round then this part of the reporting is much easier. Gathered logs are investigated to determine the reasons of the failures that have occurred. The tester must find out which of those were caused by the actual errors of the device and which were not. Some of the failed iterations might be related to the automation tools or environment problems.

Once an error is found, the tester takes notes describing what has happened and in what circumstances. Those notes are very useful at later time when the new errors reports are created.

Not always the cause of an error can be easily found. Sometimes it is also reasonable to check the logs of the test iterations that have been executed right before the one that has failed. Some additional clues may be found.

7.2 Result processing

If during the testing round the execution was interrupted then results may be in several pieces. To make the study of the logs easier it is good to join the separate parts into one continuous record.

A problem with the device can occur at any test step of the test case. If it happens at the very beginning then it might be so that a vast part of the video footage that is being taken does not provide any new information about the problem. Some test cases, those which use network connection in particular use long verification times. Hence, the video log can be quite long. To make the video navigation easier and to reduce the size of a log file, the video file has to be shortened. It is especially important if the file needs to be sent over the internet or uploaded to some error data base.

7.3 Creating error reports

If the failures turn out to be caused by the problems with the devices software then the new error report has to be created. Based on the logs created during the testing the description of the error is prepared. Apart from the information about the issue also the information about the environment and tested device variant has to be included in the report.

If the error report for the specified problem already exists, then there is no need to create a new one. It does not mean that nothing should be done. Providing people who are responsible for fixing the errors with all possible information is important. The fact that a certain flaw has been reproduced on specific build is important. Even if the error may seem to be the same it may happen that the cause of it is totally different. It is up to errors handling team to decide whether to create a separate error report or to merge both cases.

It is important to understand that errors reported are usually not easy to reproduce. Even though that same test steps are used. There are many applications and services running on the device at the same time. Interactions between those can be very complex. The total number of possible “states” in which a tested appliance can find itself is uncountable.

To help developers responsible for the specific application to fix the problem it is of vital importance to describe all circumstances in which the reported error occurred. The same error reproduced by performing different test steps may give them a clue on what is the cause of it. Unfortunately that kind of information is not always available. Sometimes the problem that was noticed does not seem to have anything to do with what the device was “doing” at the moment when it occurred.

7.4 Filling in execution report

Creating the execution report starts with filling in the details about the tested product. Those details include matters like hardware version, software build

and the amount of data that has been put on a device's memory. The next issue is to provide the information about the testing environment. This section of the report contains details about the automation tool's software version, type of connection that was used to connect the device and any additional equipment which has been utilized during the testing round.

The most important part of the report is the findings and faults section. Any errors and problems that have been identified during the execution are highlighted here. Each error is briefly described and an error report id provided. The error id comes from the database to which the issue has been reported.

The same error may have occurred repeatedly during the whole execution. There is no point in attaching the same information several times to the report file. The error should be mentioned only once. All the excess data about the error is discarded from the official report document.

The execution report is finalized by adding the tester's own comments on the product's shape. Comments can be anything that the tester may find important to mention, but at least they should contain information about the overall reliability of the device, seriousness of the errors that have been found (according to the tester).

It is important to understand that the execution report is not meant for the team responsible for fixing the error, it is for the management staff, which will make decisions based on the results of the testing. Technical details should not be attached to the report. The report should give the reader good image of the product's current maturity state.

7.5 Verifying errors

An error which has been fixed usually requires verification. Often a request to retest the same functionality with newer software build is being issued. It means that the tester must do the testing using the same test steps that were

originally used when the error occurred and check if the problem still exists. In some cases the tester might be requested to use some additional, special procedures that will make the verification even more reliable.

That kind of request may be issued couple weeks or more after the error was originally reported. Sometimes some additional information is needed about the error. In the meantime the tester might have performed many other testing executions. To avoid confusion and mistakes it is important to make as good notes about the error as possible. Those notes should also contain precise information about the software build, hardware version that was used and other relevant matters. Otherwise it might be difficult to recall what exactly happened when the error had occurred. It is also important to preserve the results and self made notes for some, reasonable time.

The idea of the error verification is to find out whether the applied fix really works. It might happen that the error is in a different place or that the fix only partially fixes a bug. The frequency of consecutive malfunctions may be decreased, but the error might still be there.

Different products may share some software elements. Hence, in some cases a request to verify an error on some device is not a direct result of an error report created for that product. Instead the error verification is done to make sure that the same problem is not found in the device, which is a “close cousin” of the product from which the issue was actually found.

8 RESULTS

During the writing process of this thesis many new, automated testing scripts were created. They were successfully used to perform multiple automated reliability testing executions on several, different products. However, there are still few challenges, which need to be addressed to make the automation even more useful.

The tester does not have a full control over what is happening on the device. Mobile appliance is a connected type of device. It means that it depends on the resources that are not or not fully controlled by the appliance. This makes measuring the reliability of the device more difficult, because the tester must take into account that there are external factors that affect the testing process. For example, network conditions can change during the day depending on how heavy the internet traffic is. This can affect the maximum network speed and thus change the web pages loading times, performance of the video streaming server and others. It may cause test iterations to fail even though no problem with the device occurred.

The simplicity of the scripting language used for creating the test cases makes it easy to learn and use. However, it has also its disadvantages. In some cases it can be slightly stiff. The designer of test cases may sometimes run across some problems when implementing the desired behaviour. Usually it can be implemented somehow, but the designer may be unhappy with the solution, knowing it could have been done much better if only few more options were available.

Thanks to the study of the automation tools, several flaws were found in them. Those were reported and appropriate fixes have been released. Apart from that, few suggestions on how to improve the usability of the automation tools have been made.

The testing instructions have been updated with new findings that were noticed during the study period. Those include matters like good practices

when creating new test cases, advices and clues on how to create good test scripts, common problems that occur with automation tools and others.

Some improvements were also made to the reporting process. The amount of logs that need to be processed during the reporting part of the testing is sometimes very large. The necessary information can be scattered around many files. To facilitate the retrieval of specified information, a new tool was developed. It allows processing of many files of the same type simultaneously, obtaining the needed data and presenting it in a convenient way. Another tool that was developed can be used to retrieve the information from several different sources and join them logically. There is still room for more tools that would make the reporting process easier.

During the study some ideas on how to improve the testing laboratory were proposed. Few of those were positively welcomed and implemented. For example, new way of supporting more environment configurations without adding new hardware was proposed.

9 DISCUSSION

The main goal of the project was to study and improve the process of automated testing of mobile devices. I think the goal was successfully achieved. During the process of writing this thesis I have managed to improve the testing scripts that are used for automated, reliability testing of many different products. I have learnt how the automated testing should look like and what has to be considered when deciding on such an approach to testing.

There are many areas of non-functional testing. During the study I have learnt that not all of them are easily automated. Some of those areas are not automated because the gain is not significant. In some cases the workload needed to automate the area of testing may even exceed the benefits. This is an important observation. Even though automation is very useful, it is not a silver bullet. It will not solve all of our problems and we should not be too persistent in our attempts to automate everything. Instead, we should consider it as one of the tools we have and use it when it is suitable.

I have now a much better understanding on what kind of software is vital for the mobile device and how to properly test it. I am now familiar with typical problems that may occur when automation tools are utilized and how to solve those. Thanks to the study that I have done for the purpose of that thesis, I am now better prepared for new challenges that I will need to face in future.

When I was working on the instruction improvements I noticed several times that something that used to be true is not true anymore. In some cases it is true only under some conditions. That thought me how liquid the matter is. In research and development projects it is not possible to establish one optimal way of doing work. Such projects by their nature are changing constantly, especially in mobile devices industry everything is changing very rapidly at the moment. To maintain good quality and efficiency, improvements have to be made all the time. For the instructions to be good they have to be either very general or contain many product’s specific chapters.

There are, however, matters that do not change. Like in most projects the passing of information is a very important. There are many members in the testing team. A mailing list could be used to inform others about our findings. This works fine if the information is needed only now. However, if it is something that should be preserved and easily accessible then it is not a good solution. Everyone receives a lot of mails every day. Sooner or later the information is lost somewhere in the mails archive. One way to do it properly is to use wiki pages. Wiki offers a collaborative way of maintaining the instructions. Everyone can add and modify pages. Information is easily accessible; everyone has access to the newest possible version.

By doing the testing I familiarized myself with the automated testing environment. I learnt how it is designed, how to configure it and how to utilize it in practice. I also learnt what its strengths and what its limitations are. Those tools are in constant development, just as the new products are. To keep them in good shape the tool team is working constantly on the improvements. The testers, who use them, are also responsible for maintaining good quality of the tools. Their duty is to report the problems that occur. It is also in their best interest to do that because, after all, the problems with the tools affect them the most.

Being on the cutting edge of technology has its price. All the people working for the project have to be agile. They need to be able to quickly adapt to the changing environment. The most important of their characteristics is that they need to be good team players. A major project like the creation of the new mobile device requires full cooperation from all the parties involved in the process. That is I think the most important lesson I have learnt as testing project member. I am sure it will help not only in my work, but in life in general.

Quality Assurance (QA) gains more and more importance as the complexity of mobile devices increases. New sophisticated functionalities are introduced in every new device. To provide quality products, more and more testing has to be done before consumers can buy products at a shop. What is quality, what does it mean that a product has a good quality? There is no single answer for that. Different persons may perceive it in different ways. There are many

elements which affect the "feeling" of good quality. Not everybody will appreciate the effort that was put in polishing every single element; still work has to be done because every left out element can destroy the users’ feeling about the quality of the whole product.

REFERENCES

1 Chip Design. Referred to on November 30, 2009 http://www.chipdesignmag.com/denker/wp-content/uploads/2007/08/cost-of-change.JPG

2 DIGIA. Referred to on November 30, 2009 http://www.digia.com

3 Gao, J., Tsai, H.S., Wu, Y. 2003. Testing and Quality Assurance for Component-Based Software. Artech House, Incorporated

4 Merriam-Webster Online Dictionary. Referred to on September 19, 2009 http://www.merriam-webster.com/dictionary/quality%20assurance

5 Test method (ASTM definition). Referred to on November 20, 2009 http://en.wikipedia.org/wiki/Test_method

6 The History of Computing Project. Referred to on September 17, 2009 http://www.thocp.net/timeline/1672.htm

7 The OPTISM WWW site. Referred to on November 25, 2009 http://www.cc.ysu.edu/~rlhoover/OPTISM/reliability_validity.html

8 The Testing Standards Working Party. Referred to on November 25, 2009 http://www.testingstandards.co.uk/definitions.htm

9 Watkins, J. 2001. Testing IT : An Off-the-Shelf Software Testing Handbook. Cambridge University Press