OULU UNIVERSITY OF
APPLIED SCIENCES

Juha-Matti Suomaa

THE TEST AUTOMATION IMPLEMENTATION FOR A JAVA-BASED ENVIRONMENT

THE TEST AUTOMATION IMPLEMENTATION FOR A JAVA-BASED ENVIRONMENT

Juha-Matti Suomaa
Bachelor's thesis
Fall 2015
Information Technology
Oulu University of Applied Sciences

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietotekniikan koulutusohjelma, Langattomat laitteet

Tekijä: Juha-Matti Suomaa
Opinnäytetyön nimi: Testausautomaation toteutus Java-ympäristöön
Työn ohjaajat: Pete Pietilä, Mikko Suhonen, Riitta Rontu
Työn valmistumislukukausi ja -vuosi: Syksy 2015 Sivumäärä: 31

Työn tilaajana toimi Oy LM Ericsson Ab:n Oulun yksikkö. Työn tarkoituksena oli toteuttaa yksikön sisäisen testiautomaatiojärjestelmän testitapaukset uuteen, koko yrityksen laajuiseen testiympäristöön sopivaksi.

Käytettävä järjestelmä oli jo laajassa käytössä yrityksen muissa yksiköissä ennen kuin se otettiin käyttöön Oulun yksikössä. Järjestelmä sisältää useita avoimen lähdekoodin työkaluja, joiden avulla testitapauksia kehitetään ja suoritetaan.

Testitapausten perustoiminnallisuus saatiin toteutettua uuteen ympäristöön aikarajan puitteissa. On kuitenkin mahdollista, että joitain yksittäisiä asioita jäi huomioimatta, sillä loppuvaiheessa testitapauksia ei ennätetty testata käytännössä niin täysmittaisesti kuin oli aikomus. Mahdolliset ongelmat kuitenkin tulevat varmasti testatessa ilmi ja ne on mahdollista korjata jälkeenpäin. Testitapaukset ovat rakenteeltaan modulaarisia, joten niitä voidaan tulevaisuudessa uudelleenkäyttää ja laajentaa tarpeen mukaan.

Asiasanat: Ericsson, WCDMA, Java, testiautomaatio

ABSTRACT

Oulu University of Applied Sciences
Information Technology, Wireless devices

Author: Juha-Matti Suomaa
Title of the bachelor's thesis: Test Automation Implementation for a Java-based Environment
Supervisors: Pete Pietilä, Mikko Suhonen, Riitta Rontu
Term and year of completion: Fall 2015 Number of pages: 31

This thesis was commissioned by Oy LM Ericsson Ab Oulu site. The purpose of the thesis was to implement in-site test automation system test cases compatible to a new company-wide test environment.

Then new environment had already been used in other sites of the company before it was taken into use in Oulu site. The system contains lots of different open source tools to develop and run test cases.

The general functionality of test cases was achieved within the given time limit. However, there is a little possibility that some individual points were left unnoticed, because testing of cases was not so comprehensive in the final moments as it was meant to be. However, possible problems will pop up during actual test runs and then it is possible to fix those problems afterwards. Test cases are structurally modular and reusable. Additions and modifications can be easily implemented later if needed.

Keywords: Ericsson, WCDMA, Java, test automation

PREFACE

This thesis was made during spring and summer of 2015. The thesis was commissioned by Ericsson Oulu site. The aim was to implement test cases to company-wide test automation environment.

The months working for Ericsson during the thesis work and before that have been very rewarding. I want to thank all the co-workers I had possibility to be part of this weird bunch of people. I never lacked help or ridiculous nicknames. Special thanks I would like to grant to my supervisors Mikko Suhonen, Pete Pietilä and Riitta Rontu for guidance.

Oulu, August 2015

Juha-Matti Suomaa

CONTENTS

VOCABULARY

3GPP	Third-Generation Partnership Project
ADB	Android Debug Bridge
CN	Core Network
CS	Circuit Switch
EUL	Enhanced Uplink
FTP	File Transfer Protocol
HSPA	High-Speed Packet Access
HSDPA	High-Speed Downlink Packet Access
HSUPA	High-Speed Uplink Packet Access
IDE	Integrated Development Environment
OOP	Object-Oriented Programming
OS	Operating System
POM	Project Object Model
PS	Packet Switch
RAB	Radio Access Bearer
R&D	Research and Development
RDP	Remote Desktop Protocol
RF	Radio Frequency
RX	Receiver
RNC	Radio Network Controller

SIM	Subscriber Identity Module
SSH	Secure Shell
TX	Transmitter
UE	User Equipment
UMTS	Universal Mobile Telecommunications System
WCDMA	Wideband Code Division Multiple Access
XML	Extensible Markup Language

1 INTRODUCTION

This thesis was commissioned by Oy LM Ericsson Ab's R&D site in Oulu. This Swedish company was founded in 1876 and its global headquarters is located in Stockholm, Sweden. Nowadays, the company have almost 120,000 employees around the world, of which 25,700 are R&D employees (1). Oulu site was founded in 2012 to develop small cell RBSs for Ericsson wireless network products portfolio.

The aim of this thesis was to expand the already existing Oulu in-site test automation environment to work in a new company wide Java-based environment. The test automation environment was developed on top of commonly worldwide used test automation open source components. The existing system had been noticed to have some limitations so it was a reasoned decision to replace the test automation with a new environment.

The previously implemented test automation system was operated with shell scripts on a Linux server. The upcoming test automation system runs automated test cases using Java based scripts.

The author of this work has been working earlier for Ericsson as a project trainee. After getting experienced with testing and the used environment, it was a natural decision to continue doing the thesis for the company while a possibility was offered. The job description has changed from testing towards development, and development has brought its own challenges as the author is not very experienced with a shell script or Java languages.

The challenges to accomplish the thesis work within a time limit consisted of understanding shell scripts of the existing test automation and moving these test cases to a new environment. New test cases should at least fulfil all requirements of the old environment.

2 OVERVIEW OF TERMINOLOGY AND NETWORK TECHNOLOGIES

This thesis has a lot of terminology and abbreviations thus it is worthwhile to make a brief overview of the basics before introducing the actual work. During the earlier projects at the company, the author has acquired most of the following information that consists of testing small cell technology and creating descriptions of test environments.

Since the main focus in this thesis is in the development process, it is not necessary to cover network features at a detailed level. Main issue is to concentrate on those features that are justified and occur during a test case development.

2.1 WCDMA

The basic architecture of a Wideband Code Division Multiple Access (WCDMA) network consists of following major elements: Radio Base Station (RBS), Radio Network Controller (RNC) and Core Network (CN). Core Network can be divided into Packet Switched (PS) and Circuit Switched (CS) cores. Beside the mentioned parts, User Equipments (UE) as network clients are relevant for the network existence. A simplified illustration of a WCDMA network is presented in Figure 1.

RBS refers to a NodeB or an eNodeB and it is the part of the network communicating directly with mobile devices. NodeB is a term that is used when speaking about a 3G technology. An eNodeB is an evolved version of NodeB and it refers to a 4G technology. To put it simply, both of these technologies can be called nodes. But because the focus of the thesis is in the WCDMA network, from now on a node refers to a NodeB.

Nodes have different configurable features, for example transmitter power and parameters of transmitted channels. RNC is a network element that is used to govern these features on RBSs that are connected to it.

UE is a mobile device that is used to connect to the network via RBS. UE may refer to a phone or a tablet computer including a SIM card, but it can also be a USB-modem a.k.a. dongle depending on the subscriber's requirements.

RNC monitors RBSs and is able to get information of the UEs that are connected to RBS and perform a handover between cells if necessary. The handover is a situation where UE is moving within a range of two or more cells. If one of the neighbor cells, where UE is not connected, has a better signal strength, RNC transfers a connection to other cell providing better circumstances for UE.

Core Network is part of the network where the end user data is transmitted between devices via a high speed physical connection. Usually optical fibers are used. As mentioned earlier, core network can be divided into two parts, which both perform different tasks. A circuit switched core is responsible for basic voice calls, while a packet switched core handles a data transfer.

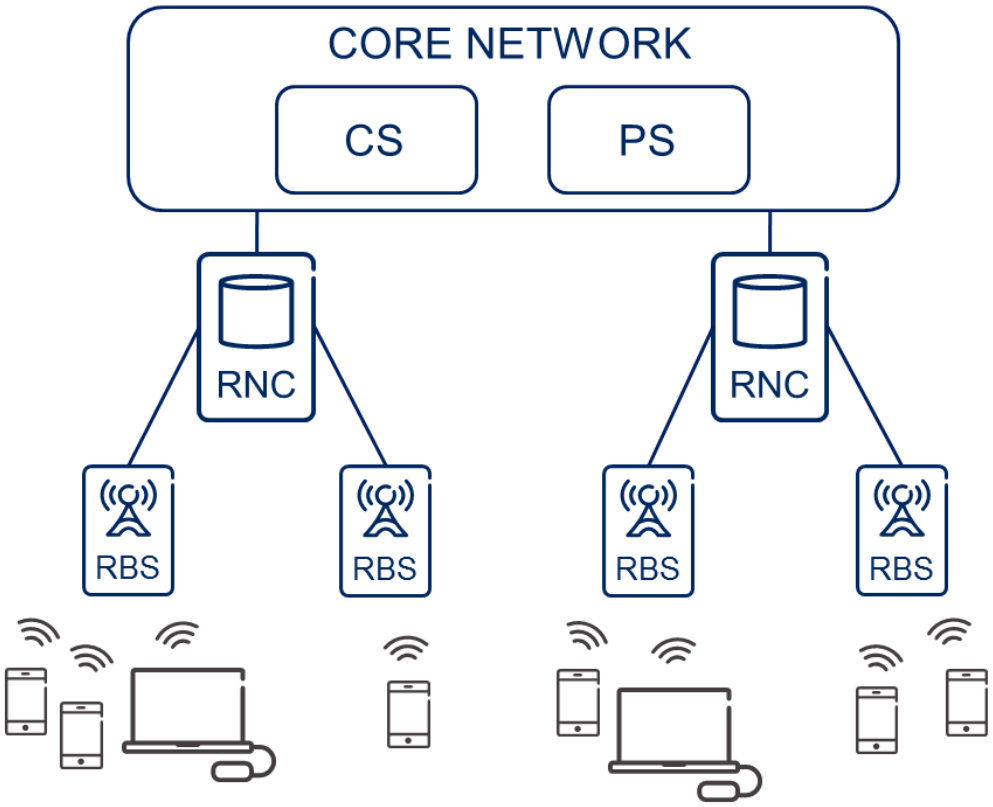

FIGURE 1. Architecture of a WCDMA network

The Third-Generation Partnership Project (3GPP) is a body of several standard development organizations that specifies 3G and GSM systems (8). During the evolution of WCDMA, there have been several 3GPP releases and every release has its own specifications. The test cases explained in chapter 5 contain tasks where data rates are compared to predefined values. These values are picked from the following releases.

Release 99

The third generation telecommunications technology was released in 1999. This R99 (Release 99) called the first UMTS (Universal Mobile Telecommunication System) release was able to achieve a data rate up to 384 kbit/s for downlink and uplink.

HSPA

High-Speed Packet Access (HSPA) is an upgrade to WCDMA network that consists of High-Speed Downlink Packet Access (HSDPA) and High-Speed Uplink Packet Access (HSUPA), also called Enhanced Uplink (EUL) (7, p. 11).

HSPA was an answer for increased data service demands after Release 99. Depending on the used network solutions, HSPA's theoretical maximum downlink data rate can be up to 21Mbit/s with a single carrier technique. Nowadays, later released WCDMA releases with multiple carriers and antennas provide theoretical data rates up to 337Mbit/s (9).

2.2 Small cells in heterogeneous network

Small cell refers to a low-powered base station that features a relatively short signal range. When the range of macrocell in a sparsely populated area can be even tens of kilometers, the range of picocells and femtocells is only from tens of meters to hundreds of meters. Cell size differences of RBSs are presented in the following figure. (Figure 2.)

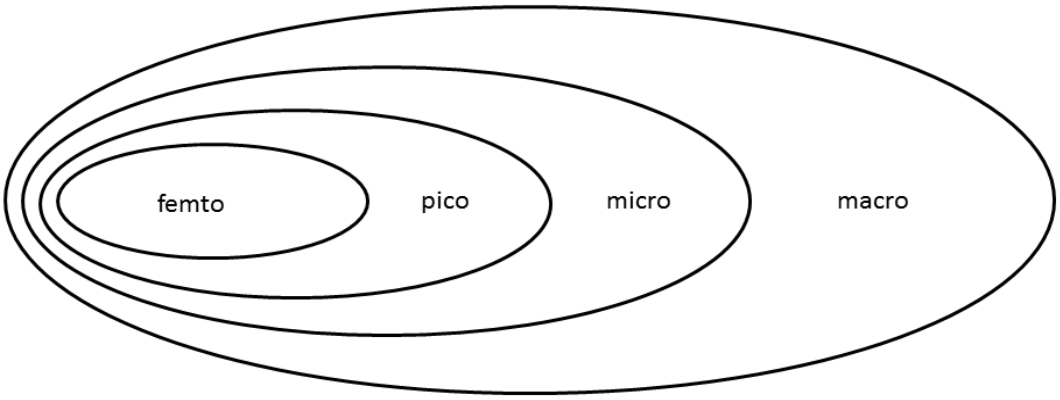

FIGURE 2. Demonstration of RBS cell size coverage area

Heterogeneous network, also called HetNet, refers to a network where multiple types of nodes are used to achieve the best possible coverage. Small cells are suitable for areas where the coverage might be poor, like indoors, or it is not reasonable to build an expensive and power-consuming macrocell. Deploying small cells to a macrocells coverage area increases performance of macrocell by offloading traffic to a small cell coverage area while it also reduces the power consumption of macrocell (2).

3 CONTINUOUS INTEGRATION

Usually, Research & Development (R&D) consists of multiple testers, developers and other R&D related employees. When several developers have their own section in a project, the combining of the code might turn troublesome, especially if it has been a long time since the last software integration was made.

The continuous integration is a way of software development where each developer integrates their code frequently and often to shared primary code also called mainline. Of course, developers have to be sure that their part of code is unbroken thus verification is essential before merging the code.

The true functionality of the code will not appear until the testing phase with the actual hardware, in this case, with the node. If a new software becomes on a daily basis, the test automation is really needed to save human resources to other tasks.

The great benefit of the continuous integration is that it is easier to track the overall progress of project. Daily tests reveal problems relatively quickly and developers can start immediately to hunt the problem.

When an integration interval is short, a developer will know much better what part of the code is working and what is not. Bugs never disappear, but with CI the amount of bugs are reduced.

CI and test case development

A similar way of working also applies to developing test cases since environment, libraries and test cases have several authors. Although the scripts are not used in any commercial product, it is important to notice that test cases and common libraries have other users and one bad modification might complicate their work, too.

4 TEST ENVIRONMENT

An automated test script development environment contains the WCDMA network elements mentioned in chapter 2, but also various development tools and software that are used to get tests running automatically.

4.1 Description of environment

There are many kinds of connections between development environment components. Figure 3 below introduces the basic structure of the environment. Beside the introduced components the environment contains basic network components, for example ethernet switches and routers not included in the Figure 3.

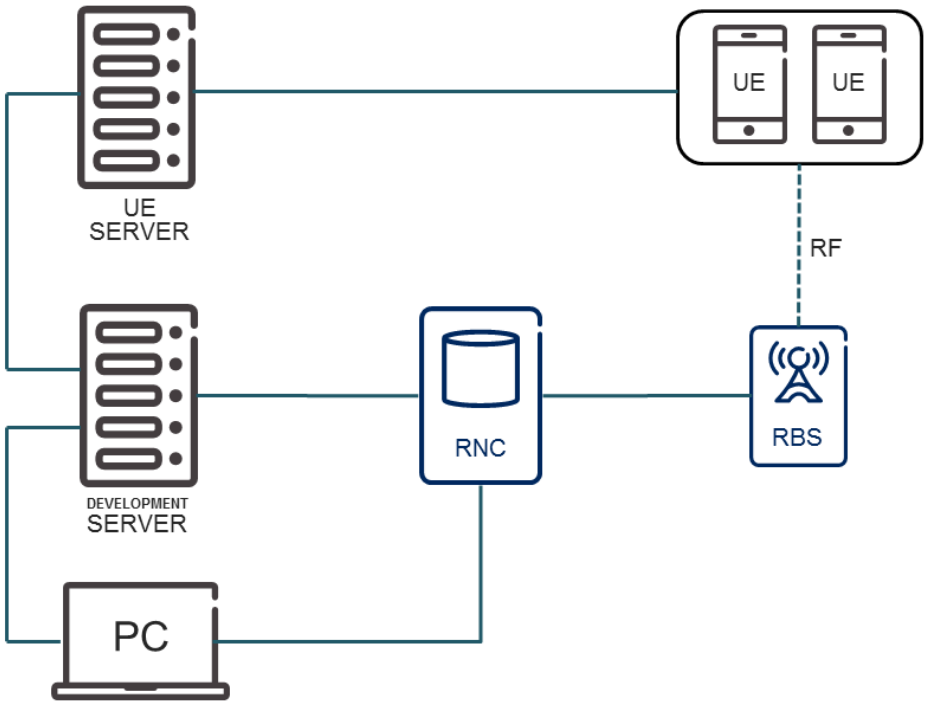

FIGURE 3. Development environment

4.2 Development server

The development server is the computer where scripts for the test automation are created. The server is running a Linux operating system that is installed with all necessary software packages for the test case development and has all the connections configured to run the finished test cases.

For development and testing Ericsson uses their own test framework that consists of an open source software with modifications. Modifications are held as minimal as possible so that the compatibility to open source software by different development groups around the world remains as good as possible.

Eclipse

Eclipse is an open source IDE (Integrated Development Environment) that can be used to develop the software in various programming languages, but mostly it is used for developing Java-based code (3). Typically, an IDE application consists of a source code editor, a debugger and build automation tools. Eclipse can be customized by each user and can be supplemented with various tools.

When Eclipse was opened for the first time, the software asked the user to select a workspace. Workspace, also called working directory, is a folder where upcoming projects are saved. During the Eclipse configuration multiple plugin installations should be done to define necessary network connections. Installed plugins are described below.

TestNG

TestNG is a testing framework for the Java programming language that is available as a plug-in for Eclipse. TestNG allows running tests straight from Eclipse and it monitors output during test execution.

TestNG offers a possibility to use annotations. Annotations are set before methods and depending on a given annotation, tests will be executed in a certain way. For example, with the @BeforeTest annotation it is possible to define a method that will always run before other methods. Correspondingly, using the

@AfterTest annotation will run a method after all other methods have been executed.

It is possible to add test steps inside test cases. With steps it is easier to divide a case into logical parts that happen during the test case execution. For example, if the parameters of channels have to be changed before the transmission, it can be done inside one step. Test steps also help examining logs afterwards. In the beginning of the test step the wanted title can be written for the step so that it later appears in a log file. Annotations and test steps are presented in the following code example.

```
@BeforeTest
public void beforeTest(){
        beforeMethodExample
}

@Test
public void Test(){
        setTestCase(testId,description);
        setTestStepBegin("Test step 1");
        testExample
        setTestStepEnd();
}

@AfterTest(){
        afterMethodExample
}
```

Maven

Maven is a build automation tool running on Eclipse. Its main aspects are describing dependencies and how software is built in a project. The core of the project configuration is a pom.xml (Project Object Model) file. This file includes all necessary configuration and dependencies for running code and other software with Eclipse. Following script shows an example of how a TestNG plugin can be defined in a pom.xml file.

```
<dependency>
        <groupId>org.testng</groupId>
        <artifactId>testng</artifactId>
                <version>6.8</version>
        <scope>compile</scope>
</dependency>
```

Git

Git is an open source version control system from small to large projects that is operated from a command-line. The basic idea of Git is that several authors of a project have their own workspace and local repository that is a copy of a remote repository where all authors are connected. Authors are able to edit their own version of the project as much as they want and only if desired, authors can merge an edited project to the remote repository.

The process of getting the wanted project files from a single author to the remote repository consists of few steps. The actual software development takes place in a working directory where files are copied to the index with an add command. The index is a storage for files that are waiting for a commit command to move files to a local repository. The last step is to push files from local repository to remote repository. Since all project authors have possibility to push files from their local repository to remote repository, it is important to use caution especially if common files to all are modified.

There are two ways to get files from the remote repository. A fetch command can be used to get files to the local repository while pull command merges files directly to working directory. Before a pull command, it is recommended to use a stash command to temporarily save current working directory. If something goes wrong, with stash pop command it is possible to return the old working directory contents back. The Git progression is illustrated in Figure 4.

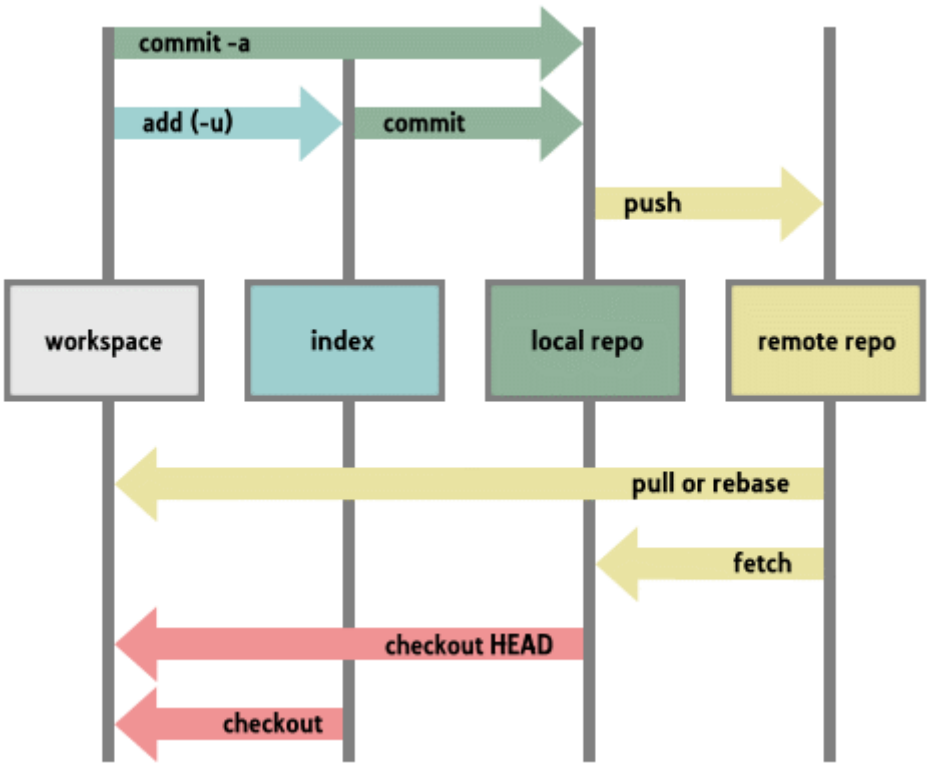

FIGURE 4. Git progression diagram (6)

4.3 UE server

UEs are controlled with a Windows server, which is capable of running several phones and dongles simultaneously. The server was installed with a test environment compatible UE controlling software that could be run from GUI (Graphical User Interface), command-line or remotely via Eclipse running on the development server.

The software is used to perform all the tasks associated with the UE usage and it also provides information about UE's status during different situations. The software is capable of initiating a voice call and data connection, which are needed for PS traffic test cases. The software monitors the data rate during the FTP transmission and in the end of a download or upload it calculates an average throughput.

Usually the UE controlling software is controlled via Eclipse, but for debugging and development purposes a direct connection is sometimes needed. A direct access to the UE server is established with a remote desktop software from the development server.

4.4 User Equipment

In regular daily use phones and dongles are used over the air interface but in this environment UEs are connected to an RBS via an RF cable. The main reason for this solution is that a straight cable connection to a node prevents the used UEs to connect to unwanted nodes and the used node does not disturb other devices. With an RF cable it is also possible to get stable results without interference from air interface that causes signal weakening and reflections from surrounding objects. The actual interference testing is carried out separately.

Selected UEs

The requirements of test cases determine which UEs are suitable for use. For a pure data transmission it is recommended to use USB-modems. These are designed merely for data transmission in mind, and thus they might be slightly more efficient than an ordinary phone that is also used for voice calls. Some cases contain data transmission during a voice call and this time it is necessary to use phones instead of USB-modems.

For data transmission there were available three different dongle options, which were Huawei, Quanta and Netgear. From these alternatives only Quanta and Netgear were suitable for the environment used in this thesis work. Huawei was not compatible option due to a couple of reasons. Firstly, it did not meet certain network requirements that need to be tested in later implemented test cases. Secondly, the UE controlling software did not have support for this UE, thus the controlling of Huawei would have been impossible.

For test cases containing voice calls the only available option was Samsung Galaxy S3 mini. The phone is running an Android operating system and it is compatible with the UE controlling software.

Because the UE server is attached with several UEs, it is necessary to be able to check before the test case what kinds of UEs are attached. The UE's ID is one way to check if an attached UE is a phone or a dongle. Android phones use an ADB (Android Debug Bridge) tool that adds an extra prefix before an actual ID number. By recognizing that prefix it is possible to find a voice call capable phone among all UEs.

Shielded box

UEs are kept in an RF shielded steel box with pull-throughs for RF and USB cables. Specially designed shielded boxes prevent signals moving over the air to disturb the UEs of surrounding test environments and also reduce the signal strength from possibly leaking RF cables attached to UEs inside the box.

4.5 FTP server

FTP (File Transfer Protocol) is a network protocol that is used to transfer files between systems through the network. In this environment an FTP server is used to download and upload files of different sizes to measure the throughput during the data transmission.

5 TEST CASE DEVELOPMENT

The existing test automation environment is operated with shell scripts. The environment has some limitations, so it will be replaced with a new system where test cases are programmed with the Java language.

5.1 Object-oriented programming

Because the new environment is based on Java, it is capable of using classes and objects. This is called Object-Oriented Programming (OOP). Classes and objects have certain features that are explained in this chapter.

5.1.1 Class and objects

Class refers to a code structure that contains definitions for certain case. It is a template to create objects that have something in common with each other. As a practical example, every animal has legs, age and color but the number of legs or age and color varies between animal species.

Objects are structures where values that are initialized in a class can be taken in use. As a reference to the earlier mentioned animal example, an object may be a cat that has four legs, certain age and black fur. In other words objects have states. States are stored to variables, which can be for example a string, an integer or a Boolean.

There is an inheritance between objects. Practically, this means that it is possible to call other objects and create relationships as long as the objects location is known. This makes it possible to reuse the once created code and save computer resources.

5.1.2 Method

Methods are structures where objects states are used to perform an operation. A method can be used e.g. to communicate with other objects, perform calculations and return values. In other words it is a crucial part of building functional test cases.

5.2 UE and Network element control

Getting access to the test environment devices requires different kinds of tools and resources. Some of these are owned by Ericsson and some are open source software.

Management tool

Ericsson have an internal command line scripting-based management tool for nodes and RNC. The original plan was to run the tool on the development server. Because of an unknown problem the tool did not run correctly on the server and it was decided to enable a connection to another server where the management tool was running without problems.

Resources

The backbone for running test cases and getting an access to all necessary components are resource files called Spring Beans. These are XML based configuration files that contain information about the environment e.g. from servers and nodes.

Basically, it is possible to gather all necessary information in one bean file, but in this environment it was decided to use several files. It does not really matter if there is only one or tens of bean files, but of course the files should be named clearly. One thing that matters is a bean id. Bean id refers to an individual component in the environment, for example a node_1 that has a unique name.

The bean files are located in a folder called src, which is an abbreviation of sources. Before the test begins to run, all bean files are scanned to find necessary resources that are needed for the test case. This is the moment when there should not be found components that possess identical IDs. If identical bean id components are found, the test cannot be run, because it is not known which one should be used. An example of nodes with separate named IDs is described in the following example.

```
<beans>

        <bean id=”nodeID class=”Nodes”>
                <property name=”nodeList”>
                <list>
                        <ref bean=”node_1”  />
                        <ref bean=”node_2”  />
                </list>
                </property>
        </bean>

        <bean id=”node_1” class=”nodeResource”>
                <property name=”name” value=”node_1”  />
                <property name=”host”  value=”1.1.1.1”  />
                <!--  additional information  -->
                <property name=”password” value=”password1”  />
                <property name=”username” value=”username1”  />
        </bean>

        <bean id=”node_2” class=”nodeResource”>
                <property name=”name” value=”node_2”  />
                <property name=”host”  value=”1.1.1.2”  />
                <!--  additional information  -->
                <property name=”password” value=”password2”  />
                <property name=”username” value=”username2”  />
        </bean>

</beans>
```

Resource Manager

The company has its own common libraries where most of the resources for controlling the nodes and also other network elements can be found. If it is known what kind of node is used, it is not necessary to reinvent the wheel, but to just find the correct resource.

In a bean file it is possible to refer to the class where the resources for controlling the node are found. An example of referring to the class can be seen in the script above where both nodes refer to a nodeResource called class.

5.3 Test cases

The existing test automation system is running different kinds of test cases. For this thesis it was chosen to develop traffic test cases with the help of existing shell scripts. The chosen test cases compile a solid test set that is possible to accomplish within the time limit.

Traffic test cases measure the data speed in the network. Practically, this means that during the data transmission samples are taken at certain intervals. After the test case is completed, the samples are averaged and compared to the pre-defined value. List of chosen traffic test cases is shown below:

- R99 Downlink
- R99 Uplink
- R99/R99 Uplink + Downlink
- HS Downlink
- EUL Uplink
- EUL/HS Uplink + Downlink
- EUL/HS Uplink + Downlink, 2 UEs
- CS + R99/R99
- CS + R99/HS
- CS + EUL/HS

5.3.1 General features in test cases

There are certain things to be checked and done in the beginning of every test run. Here are explained some basic things that are relevant to get test cases run smoothly without problems caused by a bad test automation script. More individual features are explained in chapters 5.3.2 and 5.3.3.

Node status

The first things to check is that the node is up and running, because without it there is nothing to test. If the node is not found, the test case is stopped immediately. If the node is found, the test keeps running. It is useful to print out an id

of the used node. This is useful if several nodes are used and the logs have to be examined later.

Transport channel adjustment

In the beginning of the test cases a simple if-condition or assertion should be programmed to check what channels are enabled on the RNC. The if-condition is a very common statement in many programming languages and the assertion is a TestNG method that can be used to compare values. The following script gives an example of a situation where channel states are explored, changed if necessary and asserted. If the assertion is failed, the test case keeps running forward.

```
if(!channels_enabled){

        enable_channels.send(”activation command to RNC”);

        assertTrue(”Activation failed!”,
        channel_states.contains(”channel_disabled”));
        //check channel and print message if channel is still disabled

}
else{

        log.info(”Channels OK”);
        //print information to log file
}
```

If certain channels are enabled, but should not be or vice versa, this might lead to a fake results or a test case failure. For example, if EUL/HS features are disabled, it is impossible to obtain a data rate of HSPA requirements.

Timeout

Every test case is required to have a predetermined timeout. Sometimes there could be a situation where the test case gets stuck in an infinite loop and stops only when the test run is shut manually. A predetermined timeout makes it possible to exit the loop automatically after the time has elapsed.

There could also be a case where nothing goes wrong with the script, but the data rate is just too slow. When the test case has a predetermined minimum data rate, it can also be measured a rough time limit. By knowing this time limit the timeout can be set, after which there is no reason to run the test case anymore, because the case has already failed due to a low data throughput rate.

5.3.2 Data transfer cases

In this chapter the first seven test cases in the list of chapter 5.2 are called data transfer cases to separate them from multi-RAB cases. Multi-RAB test cases are explained in chapter 5.3.3.

It is common for data transfer cases that all data is going through a PS core. This means that the data is sent in packets via shared communication sessions.

Before the data can be transferred, the UE has to be connected to the PS core. A connection to PS core network should be established just before running test cases. If connection is enabled continuously, the UE controlling software collects logs all the time and the daily log file size is almost 200MB. Over a time the server will fill up with files and it affects the server performance. Ensuring the connection can be done with a simple if-condition in almost the same way as in script example in chapter 5.3.1.

The basic procedure in these test cases is that the UE connects to an FTP server and depending on the test case, starts downloading, uploading or a simultaneous transmission. Depending on channels that are enabled for the cell on the RNC, the used files for the transmission can be of different size. If R99 is enabled, a smaller file size is enough to get truthful throughput results, but if HSDPA is enabled, a file size should be large enough to get the transmission time longer for realistic average throughput results.

The requirements for the passed data transmission cases are that an average throughput value is greater than or equals the pre-defined value. For R99 and HSDPA cases, both have their own pre-defined values that are compared to the average throughput value.

While trying to download a file with a Quanta dongle, the UE controlling software did not start the transmission from the FTP server. Even if the FTP server responded to a ping and the transmission with a separate FTP client software was a success, the login to the server with the controlling software was failed. The software was installed correctly and according to the instructions the UE was compatible with the software. After trying different solutions, it was decided to change the UE to another and this solved the FTP problem. The encountered problem was reported, but due to the tight schedule it was not possible to search the causes for the original problem any longer, thus it was reasonable to continue working with another UE.

5.3.3 Multi-RAB cases

Radio Access Bearer (RAB) refers to a connection between the UE and the Core network. Multi-RAB is situation where the UE is using multiple services simultaneously, for example, a voice call and data transmission.

The issues to follow during the chosen multi-RAB cases are that voice call does not hang up during data transmission and throughput value stays in a range of requirements. The requirement for a minimum average throughput remains the same as in pure data transmission cases.

5.4 Running tests

During the development phase, the test cases are run directly from Eclipse powered by TestNG. TestNG have configurations to be defined before the test case execution. The defined configurations consist of a suite file, a used node and a folder where log files are saved.

Suite file

Resource files consist of a test suite file that is written in XML. All the test cases in Eclipse are run from the suite file. With the suite file it is possible to declare the wanted classes and method groups to be run.

Following script is an example of a suite file. A listener specifies the folder where the test cases are located. After the listener comes the actual test cases. Every single test case file and method to be run from case file can be defined in there. If a method is not defined in the suite file, it will be ignored during the execution, even if the method exists in the actual test case file.

```
<suite name="Example suite" parallel="false">

    <listeners>
        <listener class-name"com.ericsson.example"  />
    </listeners>

    <test name="Basic example">
        <classes>

            <class name="com.ericsson.testExample1">
            <methods>
                <include name="Test1_name" />
                <include name="Test2_name" />
            </methods>
            </class>

            <class name="com.ericsson.testExample2">
            <methods>
                <include name="Test3_name" />
                <include name="Test4_name" />
            </methods>
            </class>

        </classes>
    </test>

</suite>
```

5.5 Test case results

If the test case has been developed with a success, it is time to run the test case. While the test case is running, the output is printed to console. The amount of information in the output depends on the solutions made in the code during development. After the test case has been finished, the output is saved to a log file.

After the test case has been executed, TestNG informs if the job has been passed or failed. The amount of executed and failed tests are counted and unveiled.

TestNG creates two different links to log files after finished execution. One is for a graphical interface used in the web browser and the other one is a copy of a console output. The graphical interface makes an investigation of logs a little easier while certain points are highlighted with colours.

6 CONCLUSION

During the test case development there were few technical difficulties. Luckily these difficulties were quite simple to fix or dodge, and they did not affect the final results, although they delayed development a bit.

Maybe the biggest drawbacks were signal leaking problems during the development. Some phones did not connect to right nodes and some did not connect at all. It was decided to separate some of the nodes to other room so that interference could be minimized.

Test cases contained the same methods between each other which enabled using the created script several times with little modifications.

Due to multiple sick leave days, the schedule went really tight and verifying of test cases were not as comprehensive as it meant to be. Due to this some little mistakes, modifications or missing parts may have left unnoticed in the final versions of test cases. Despite this, the general functionality of test cases has been found successful and the whole process has been very instructive.

The developed test cases will be in the use of other project members and it is possible to develop the cases even further if needed in the future. When the groundwork has already been done, it is faster to add more functionality in the already working code than just start everything from a scratch.

REFERENCES

1. Ericsson. 2015. Facts and Figures. Date of retrieval 13.5.2015 http://www.ericsson.com/thecompany/company_facts/facts_figures
2. Wannstrom – Mallinson. 2014. HetNet/Small Cells. Date of reatrieval 13.5.2015 http://www.3gpp.org/hetnet
3. Fowler, Martin. 2006. Continuous Integration. Date of retrieval 13.5.2015 http://www.martinfowler.com/articles/continuousIntegration.html
4. Wannstrom, J. HSPA. Date of retrieval 13.5.2015 http://www.3gpp.org/technologies/keywords-acronyms/99-hspa
5. Eclipse. Desktop IDEs. Date of retrieval 20.5.2015 https://eclipse.org/ide/
6. Lessani, B. 2012. Our Magento Git guide and work flow. Date of retrieval 15.5.2015 https://www.sonassi.com/knowledge-base/our-magento-git-guide-and-work-flow/
7. Dahlman, Erik – Parkvall, Stefan – Sköld, Johan – Beming Per 2008. 3G Evolution: HSPA and LTE for Mobile Broadband. Elsevier Ltd.
8. 3GPP. About 3GPP. Date of retrieval 8.9.2015 http://www.3gpp.org/about-3gpp
9. Thakur – Nayak – Piplewar. 2013. Evolution of High Speed Download Packet Access (HSDPA) Networks. Date of retrieval 8.9.2015 http://www.ijert.org/view-pdf/6413/evolution-of-high-speed-download-packet-access-hsdpa-networks