Institutionen för datavetenskap
Department of Computer and Information Science

Final Thesis

Improved Statistics Handling

by

David Karlslätt

LIU-IDA/LITH-EX-A--09/014--SE

2009-03-29

Linköpings universitet

Final Thesis

Improved Statistics Handling

by

David Karlslätt

LIU-IDA/LITH-EX-A--09/014—SE

2009-03-29

Supervisor: **Robert Hagman**
Ericsson AB
Site Linköping

Examiner: **Mariam Kamkar**
Linköping University
Department of Computer and Information Science

Abstract

Ericsson is a global provider of telecommunications systems equipment and related services for mobile and fixed network operators.
3Gsim is a tool used by Ericsson in tests of the 3G RNC node.
In order to validate the tests, statistics are constantly gathered within 3Gsim and users can use telnet to access the statistics using some system specific 3Gsim commands.
The statistics can be retrieved but is unstructured for the human eye and needs parsing and arranging to be readable.
The statistics handler that is implemented during this thesis provides a possibility for users of 3Gsim to present information that favors their personal interest.
The implementation can produce one prototype output document which contains the most common statistics needed by the 3Gsim user. A main focus of this final thesis has been to simplify content and format control for the user as much as possible.
Presenting and structuring information now comes down to simple text editing and rid the user of the time consuming work of updating and recompiling the entire application.
Earlier, scripts written in Perl, an iterative oriented language, were used for presenting the statistics. These scripts were often difficult to comprehend since there were many different authors with inadequate experience and knowledge.
The new statistics handler has been written in Java, a high-level object-oriented language which should better suite the users and developers of 3Gsim.

Acknowledgement

I would like to thank everyone at 3Gsim for their friendship and support. The thesis work wouldn't have been possible without the help of the following people

- Robert Hagman, my supervisor at Ericsson, for being my key consultant when making design-specific decisions. He has also provided me with information about 3Gsim whenever I have needed help.
- Mariam Kamkar, my supervisor at the University of Linköping for support during the thesis as well as help with thesis technicalities.
- Mehdi Ennajjari, colleague at 3Gsim, for helping me to communicate with 3Gsim and collecting and structuring the raw statistics data.
- Åsa Lindgren, 3Gsim Development Manager, for giving me the opportunity to work on this final thesis, for supporting me and for showing interest throughout the project.
- Everyone at 3Gsim for all their help and support.

I'm also very grateful for the feedback from my opponent Joakim Norberg.

Table of Contents

List of figures

Acronyms

- ***API: Application Programming Interface***
- ***CS: Circuit Switched***
- ***GPRS: General Packet Radio Services***
- ***IMSI: International Mobile Subscriber Identity***
- ***MSC: Mobile Services Switching Center***
- ***PDG: Packet Data Generator***
- ***PS: Packet Switched***
- ***RAN: Radio Access Network***
- ***RANAP: Radio Access Network Application Part***
- ***RBS: Radio Base Station***
- ***RNC: Radio Network Controller***
- ***RRC: Radio Resource Control***
- ***SAX: Simple API for XML***
- ***SGSN: Serving GPRS Support Node***
- ***UE: User Equipment***
- ***WCDMA: Wideband Code Division Multiple Access***
- ***XML: eXtensible Markup Language***

1 Introduction

This report is one component of the author's final thesis in Computer Science and Engineering. This master thesis has been carried out at Ericsson AB in Linköping, Sweden. The thesis work was supervised by Robert Hagman at Ericsson and examined by Mariam Kamkar at the department of Computer and Information Science at Linköping University.

1.1 Background

Ericsson is a provider of telecommunications systems equipment and related services to mobile and fixed network operators globally.
Simulation is needed to be able to fully test the equipment and some of these simulation tools are developed internally within Ericsson for strategic reasons.
One of these tools is *3Gsim* which is used in load tests of the *3G RNC* (Third Generation telecom Radio Network Controller) node. Its primary purpose is to generate load on the *Iu*[1] and *Iub*[2] interfaces but can also be used in feature and performance tests.
In 3Gsim, there are simulations of *UEs* (User Equipment), *RBSs* (Radio Base Stations), *MSCs* (Mobile Services Switching Center), *SGSNs* (Serving GPRS Support Node) and internet content providers. All of these produce statistics used for test validation.
To access information from 3Gsim, to aid the system validation, developers and users gather statistics from various sources within 3Gsim.

1.2 Aim and Purpose

The aim of this thesis is to develop a tool for handling statistics provided by *3Gsim*. The tool should gather statistics, parse and extract the sought information and prepare it for easy use by other applications. The result shall also be presented visually to the user and it's important that it is comprehensible so it aids the user to get the correct conclusion.
The statistics are used for verification of system functions and to validate performance. It also aids error detection and trouble shooting.

1.3 Method

The following activities will be performed iteratively during this project.

- Investigating need for new and improved functionality.
- Literature study to know how to implement the system and how to integrate it in *3Gsim*.
- Implementing the new functionality.
- Demonstrate the system for the supervisor.
- Evaluate and document the new functionality.

[1] ***This interface is located between the RNC and the MSC/SGSN node.***
[2] ***This interface is located between the RNC (Radio Network Controller) and the RBS node***

Domain Description

The domain description specifies the environment where this final thesis has been carried out.

1.3.1 Ericsson

Ericsson is a world-leading provider of telecommunications equipment and related services to mobile and fixed network operators globally[3]. Over 1,000 networks in more than 175 countries utilize Ericsson network equipment and 40 percent of all mobile calls are made through their systems.

1.3.2 3Gsim

3Gsim is a traffic generator which simulates network components and tests parts of mobile networks. It is mainly used to simulate components on the *Iu-b*, *Iu-c*[4] and *Iu-p*[5] interfaces. It can simulate many different nodes in telecommunications networks such as UEs, RBSs, SGSNn and MSCs. To be able to monitor all the ongoing communication, statistics is constantly logged during execution by various nodes and counters throughout the network.
Figure 1: 3Gsim Test Setup describes how a 3Gsim simulates two parts of a *WCDMA* (Wideband Code Division Multiple Access) network. 3Gsim simulates a number of RBS's and a number of UEs.
3Gsim is also used here to generate packets to put load on the network.

Statistics is commonly gathered with either the RBS or the UE as unit of interest. Statistics can be gathered per RBS cell or per user equipment as well as sum of all elements.
Other possible nodes are SGSN, MSC and user data generators.

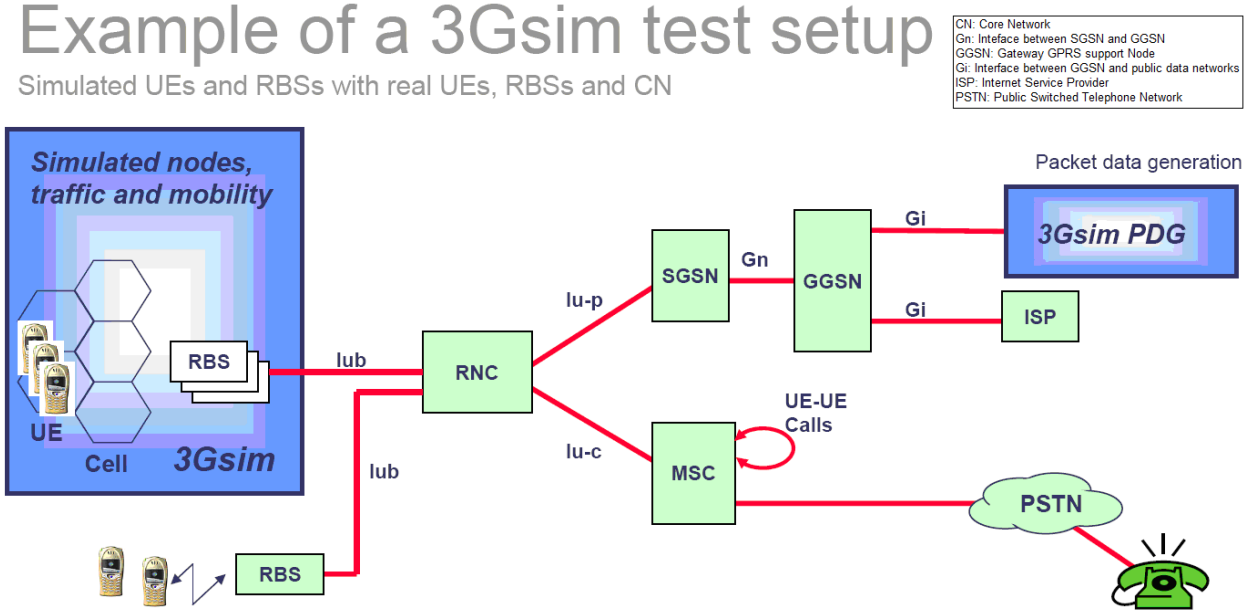

Figure 1: 3Gsim Test Setup

3 *Ericsson in Brief* [www], Ericsson AB, http://www.ericsson.com/ericsson/corpinfo/index.shtml, Retrieved 6th of January 2009.

4 *The interface between the RNC and the MSC node.*

5 *The interface between the RNC and the SGSN node.*

1.3.3 Statistics in 3Gsim

Users connect to 3Gsim through telnet and run 3Gsim commands to gather information such as counter values and behavior information.
Different counters monitoring information like, *Received bytes*, *Sent packages*, are constantly running in 3Gsim, and thus doesn't need to be started explicitly. UEs can be created within 3Gsim and those nodes send and receive data and move around in the simulated geography.
For 3Gsim users, it is today possible to get counter information for all UEs running a particular traffic behavior. Although the user can access the needed information, it is still hard to get an overview and validate the system correctly. If the user is interested in all statistics having to do with packet switched data, he probably wants to group all the statistics to compare and validate the values.
For example, when looking into information from a counter, *Received Bytes,* it would be interesting to also look at values from the counter, *Sent Bytes* to calculate packet loss at the receiving side.

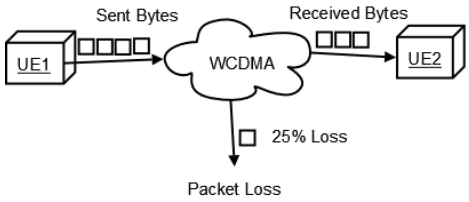

Figure 2: Counter Packet Loss Calculation

1.4 Implementation

This final thesis needed a great deal of pre studies to get a good knowledge base in order to make good design decisions. A great deal of information about 3Gsim needed to be acquired as well as getting to know the different communications methods used by nodes in 3Gsim.
The first thing to decide on was a format for how to structure the reading of the raw statistics from 3Gsim. As it was needed to parse large amounts of text, XML (eXtensible Markup Language) was a preferred suggestion. Parsing is explained more in detail in section 1.4.1.
Before starting the implementation, a format structure made in XML for the encapsulation of raw statistics, needed to be presented to 3Gsim developers.
To implement the system, developers at 3Gsim suggested Java because it's a well known object oriented high level language which is good at handling text. A *Java suite* (collection of Java classes) to structure the information was needed and the internet has many free plugins designed to handle and parse statistics[6].
If would be preferable if the XML element structure could reflect the internal data structure of the Java files as good as possible. This would enhance the parser performance and aid data extraction. Due to fixed classification of statistics, there where restrictions on the XML file. 3Gsim had specifications for structure and content for the most commonly used counter groups.
The XML structure should try to ease information gathering for these groups without compromising access to other type of information.
To access other types of statistics, a *Statistics Interface* is developed. The *Statistics Interface* provides methods to access other type of information that the 3Gsim user may be interested in.

6 ***Java & XML*, McLaughlin Brett (2000), http://java.sun.com/developer/Books/xmljava/ch03.pdf, Retrieved 24th of September 2008**

1.4.1 Parsing

XML parsing tools for Java are constantly being developed and improved. The *SAX* (Simple API for XML) allows callbacks being made during the parsing lifecycles thus enabling data manipulation and application-specific code to be inserted.
There are a number of tools that provides good SAX parsers on the internet. *The Apache Xerces* parser has very good reviews from it is users and has been developed since 1999[7]. As Xerces has such large number of users one could make the conclusion that it is a stable product and thereby suitable for this project.

1.5 Documentation

This report describes the choices and speculations of the author throughout this final thesis. The report also brings up suggestions on further updates and improvements based on the systems drawbacks.
To set standards for writing language, report structure and reference system, the *Lathund för rapportskrivning*[8] *(Reference Guide for Report Writing)* have been used as a reference. For list of references, the *Oxford System* has been used.

1.6 Research Question

The main focus of this study is not just to parse and handle statistics from 3Gsim, but to develop a general solution to group and utilize information based on their origin. The *parser content handler* (Interpreter for the XML parser) needs to manage different types of information and just consider what group the parsed element belongs to and it should handle the element accordingly. The description of the information should be free from the parser so it can be source independent and therefore be able to handle statistics from different tools.
To make it work with the statistics application as elementary as possible, it would be good to keep parsing and output configuration separated from the Java code, to rid the user of redundant recompiling. The concerns of this thesis, can be summarized by the following research questions

- How can the statistics content handler be designed general enough to handle statistics independent of source?

As mentioned in section 1.2, the main use of the statistics is in system verification, error detection and troubleshooting.

- How should gathered statistics be presented to best visualize the sought information? Can something be done to aid error detection? What improvements can be done?

7 ***Xerces*****, Apache Software Foundation, http://xerces.apache.org/ (1999), Home page visited 2008-10-20.**

8 ***Lathund för rapportskrivning*****, Andersson Ulrika, Lundquist Malin, Merkel Magnus, Önnegren Britta (2006), Linköpings universitet**

1.7 Requirements

There are three major requirements for this thesis work.

- The system must be able to gather and present information as specified by the supervisor.
- The system must provide at least one output prototype that validate the first requirement.
- The system must have an accessible interface which users can utilize to access the statistics in other ways than the prototype.

1.8 Limitations

The statistics application should only be able to handle statistics written to an XML formatted file.
The implemented prototype will only be fully tested for statistics generated by simulated UEs. Although it is preferable if the system can be prepared for handling statistics from other simulated nodes, for example RBS, MSC, SGSN and user data generators.

1.9 Target Audience

This report is primarily intended for two types of audience.

- Users of 3Gsim.
- Developers of 3Gsim.

1.10 Outline

1. **Introduction**
   This chapter introduces the thesis report.

2. **Problem description**
   This chapter describes the problem from Ericsson as well as other identified problems.

3. **Analysis**
   This chapter describes how problem solving has been performed as well as how new features have been introduced.
   It also explains the output solution, provided by the statistics application. It describes the different sub elements of the output, as well as how changes can be made to it.

4. **Results**
   This chapter describes the resulting application and what the statistics application can perform.

5. **Discussion and Conclusions**
   This chapter discusses improvements in the statistics application compared to earlier statistics solutions, and focuses on benefits and if any, on drawbacks. It also discusses future enhancements.

2 Problem Description

This chapter starts with introducing the 3Gsim product, and how developers and users utilize the system today. In the end of the chapter comes a description of the problems in the current system solution.

2.1 3Gsim

3Gsim as a system is described in more detail in chapter 1.3.2, this chapter mainly focuses on the statistics part and how validation of the system is performed.
For every simulated node in 3Gsim, 3Gsim monitors and logs information about what the nodes do and how they behave within the system. This data is saved within the nodes or in parent nodes and is updated continuously throughout a simulation. To access this data, the user can log into a 3Gsim node, and run certain 3Gsim commands that gather information from all simulated nodes in the system, and return it to the user.
A problem is that the received information can be unstructured when it arrives, often as long continuing strings of various information. The information needs to be parsed and structured before it can be used properly.
Today (before this thesis), users and developers write scripts, mostly in Perl to access the data that they are looking for and to group different counters together in order to get a good overview of the simulation.

2.2 Problems in 3Gsim

Developers and users of 3Gsim today, put a great deal of time into writing and updating scripts to gather the exact data that they are after.
Users who usually work with object oriented programming languages can have a hard time making even minor updates and changes to existing scripts as most of them are written in Perl.
Developers usually reuse and update existing scripts when wanting to test new functionality. If the entire statistics handling system could be revised and be implemented using a more modularized solution in Java, it would be easier to isolate parts that often need updating and place that information in separate external files.

2.3 Testability

As explained in chapter 2.2 scripts gathering statistics are updated often. Even though many updates are minor, there is still a great risk that bugs and false data can be introduced to the system. Both Java and C++ have advanced test libraries that simplify automatic testing, ranging from unit testing to system testing.
For the inexperienced eye, Perl can look more unstructured than Java and C++ which can aggravate the work for developers and users of 3Gsim.
For a language that is hard to comprehend, it is also hard to write good and extensive tests.

2.3.1 Prototype

All statistics from 3Gsim need to be parsed and structured for input to various user applications. The gathered data must be accessible to users for easy manual reviewing as well as input for analytic tools. The application to be implemented will simply be referenced as *the Statistics Handler*, throughout this report.
The Statistics Handler should be easily maintainable as statistics from different sources will be introduced in the system continuously. Therefore one important feature is that the Statistics Handler should be general enough to handle a wide amount of information.
As the statistics need to be available for various users and applications, an interface need to be implemented to access the information.
A presentation format was also necessary. Some requirements were set for this format:

- Functions and methods to print the output document should be implemented in the Statistics Handler.
- As users of 3Gsim work with various types of operating systems, text editors and web browsers, the output document should be of a common type as well as platform independent considering visual structure and presentation.
- Some vital information should be obtainable from the output document, at least UE status information and counter information summed up for all counters of a pre-specified type.
- Counter information should be clustered into pre-specified groups to better aid the user in identifying abnormalities in the various elements of the system.
- Counters not known to any groups should be presented on demand separated from the rest of the information.

3 Analysis

The Analysis chapter reflects back on chapter 2 and how this thesis can improve the various problems described there.

3.1 Statistics Format

Today, statistics from 3Gsim are generated as raw text data, without good structural representation and the information often represented several times in the out-data collection.
Scripts for parsing and sorting the information written in Perl exist, but are far too troublesome for users to utilize in the preferred extent.
The first problem to solve was to find a suitable format to represent the raw information coming from 3Gsim.As the format should be structured and be marked with meta-data, XML was a good choice.
In the scope of this thesis was to find and design the structural format of the XML data, not to actually implement it as this requires much further knowledge about the 3Gsim architecture.
To get all the necessary information, two separate 3Gsim commands were previously needed to gather the raw statistics. One command got statistics and meta-data for counters and counter values for UEs traversing the system during simulation. The other command retrieved information about the UEs, describing their status, different behaviors and how the UE moves geographically.

3.2 Java Suite

As mentioned in section 2.2, the statistics was chosen to be represented in a suite of Java classes. A tree representation of the data was necessary considering the information structure.
This was not too easy to find in Java; most tree solutions were either binary or had some other bound representation for both leaves and structure.
As the amount of data and nodes were expected to be quite large, it was imminent that search operations should be optimal, therefore try to keep tree traversing operations as slim as possible.
One discussed model was the *MapTreeModel*[9] which takes a *Java Map* object (an object that maps keys to values) and constructs a mapped tree. The problem with the map tree is that it only maps one key to another value, which is not enough for this project as most data has a collection of sub data. To solve this problem the mapping could be made to a *Java ArrayList* of objects, thus enabling a full tree structure.

The good part about mapping every object into the statistics is that search operations can be done very fast, with worst case scenario *O(c)* (Ordo (constant)) complexity. Also, the Map class is part of the *java.util.Collection* family thus enabling various element functions like being resizable for further optimization.

3.3 Statistics Interface

To access the gathered statistics an interface has been developed. The interface can be used to access the presentation output described in section 3.6 together with a number of other access

[9] **The Java MapTreeModel takes a Java Map object as a parameter and creates a traversable tree structure.**

methods. These methods are meant to simplify further development and aid the 3Gsim application programmer to access any data he wants.
The interface has been developed to make available any type of information that might be of interest to the user, such as "get all UEs that run a specific traffic behavior" or "get the mobility behavior for this specific UE".

3.4 Scalability

One of the main advantages by writing this tool in Java is that most developers and testers that are running 3Gsim can easily make minor changes to the tool without much knowledge about its architecture. As presenting and configuration files are separated from the rest of the application, increasing the presented information can be done without major programming skills. These changes may also involve tasks such as adding new traffic behaviors or editing counter variables.
It is very easy to have the system handle statistics for new types of nodes. The Statistics Handler can already read and structure information of the other types specified in section 1.1 and 0. To optimize processing though, it would be good to create objects similar to the UE specified in 4.2.2.

3.5 Implementation Language

As mentioned in chapter 2.2 and 2.3, users and developers put a lot of time into parsing and structuring data from the raw statistics coming from 3Gsim.
In the beginning the scripts were small and a very limited amount of data was needed when validating simulation executions.
Users at 3Gsim are not used to scripting and although Perl can be easy to learn and to create small programs and scripts in, when the application starts to grow, the code can quickly get unstructured and hard to understand.
Also, users used to C++, Java and other object-oriented languages in general, as most 3Gsim developers are, can have a hard time grasping advanced Perl scripts. Even minor updates, like adding new counters to the system, can sometimes be very time consuming.
Writing the statistics application in Java has many advantages. Developers of 3Gsim are used to object oriented languages so they should in general have an easy time understanding the code, when updates are needed.
Also, as the raw statistics are presented as XML, parsing is needed. The Java language is a particular good choice to perform this task thanks to the *SAX*.
Other tools like for example *regexp*[10], which utilizes regular expressions to seek through and extract information from text files is very easy to use in Java and *Eclipse* (a multi-language software development platform) and may improve performance considerably.
This implementation is designed so that information which is changed or updated often should be done externally. These tasks include adding or changing counter attributes, or selecting which counters that should be visible for each particular simulation.
The goal is that recompiling and changing the code should only be done when adding features and similar updates.

10 ***Regular Expressions*** **(2008) [www] Goyvaerts Jan, http://www.regularexpressions.info/, Retrieved 2008-10-24**

The Java language is known to be easy to test and there are many tools available to create, run and validate automatic tests. To have automatic tests with good coverage saves a lot of time when making update and additions to the code.
As the Statistics Handler is supposed to follow development of the 3Gsim system, updates will come with short intervals. To be able to quickly validate and test that the system runs error free, as well as it prints out the correct values by a single mouse click can be very rewarding.
During this final thesis implementation, *jUnit*[11] together with *EclEmma*[12] have been used for writing automated tests. jUnit is a plugin for writing and running the tests, and EclEmma is an Eclipse plugin used to validate the coverage of the tests.

3.6 Statistics Output

The Statistics Handler is designed to extract statistics for various needs and in several structured formats. Some requirements for the output document are specified in 0
A prototype output document needed to be designed for this thesis. The proper function calls to create this document needed to be implemented in the Statistics Handler.
The chosen format for the output document was a *.txt*-document. As users of 3Gsim use various types of operational systems, text editors and web browsers, the .txt file is a robust option. It can be opened by almost any known text editor and has very limited formatting features, meaning that different systems will not interpret the information differently.
Interesting statistics is today limited to UEs and counters that are UE-related in some manner. Some of these counters handles to the same type of information and can be clustered into counter groups. More information about these counter groups are presented in the *Statistics Handler* information, especially in section 4.1.

3.7 Improvements

Some of the vital research questions in this thesis work were *"How should gathered statistics be presented to best visualize the sought information? Can something be done to aid error detection? What improvements can be done?"*

This section describes some of the possible improvements that have been recognized during this thesis work. Most of them are not to be handled during this project but as they are mentioned and validated in this document, it can facilitate further development.

11 *jUnit* (2000) [www] , Object Mentor (www.objectmentor.com), http://www.junit.org/ , Retrieved 2008-10-10

12 *Eclemma* (2001) [www], Hoffmann Marc R., www.eclemma.org , Retrieved 2008-10-10

3.7.1 Output Improvements

With statistical information structured into groups it's easier for the user to quickly identify the counters he wants to validate. If the user is for example interested in circuit switched performance he can go directly to the group "*CS Throughput Statistics"*.
The output text document provided by the statistics handler prints out all counter groups defined and described in an external XML document, *CounterGroups.xml*.
Updates to counter groups can be done by just making changes to order, structure or definitions in the CounterGroups.xml.
As an extra security and validation measure, the user can have the Statistics Handler print out all counters that are not defined in counter groups. This helps the user to keep track of newly introduced counters in 3Gsim as well as misplaced counters and wrongly spelled counter names.
The statistics output document produces only total simulation statistics, meaning that values for a counter is summed up for all UEs that utilizes that counter and run a specific traffic behavior.

4 Results

Chapter 4 describes the resulting implementation and its features.

4.1 The Statistics Handler

The Statistics Handler is a standalone Java application that structures statistical data based on XML data together using a XML description defining the data.
For 3Gsim, the Statistics Handler needs two XML documents. One document has data for all counters consisting of *counter names*, *counter values* and an *origin* for the counters which points to the utilizer of the counters. The second document describes the UEs and the different behaviors used during the test run. The behaviors are described by *type*, a *name* and an *ID*.
The UEs are described by a unique *IMSI* (International Mobile Subscriber Identity), a traffic *behavior ID*[13], a *mobility behavior ID*[14], and a number of *cell IDs*[15], traversed by the UE during execution.
Seen below,
Figure 3: Statistics Handler **Solution** describes the workflow for developers and users of 3Gsim, how the information is passed along throughout the system.
The 3Gsim user runs a Perl script called *Get3GsimStatistics* which handles communication with 3Gsim. 3Gsim returns two sets of information, one with counter specifications and counter data and another set of information with UE and behavior specification.
3Gsim generates XML files with statistics which are downloaded down to the *xml* folder in the same folder as the Perl script is run. In the *xml* collection, there already exist documents that explain the XML statistics to the Statistics Handler. The Statistics Handler gathers the information and structures it to make it more accessible for output.
Depending on what type of output the 3Gsim user wants, he can use the *Statistics Interface* to access the data. The Perl script, *PresentStatistics,* uses the interface to present the statistics to the user.
If the user runs the *printStatisticsTxt()* function in the interface, a .txt-file called *Statistics.txt* will be created in the current folder.

[13] **A traffic behavior defines what the UE do during simulation, which packets that will be sent, and calls the UE will make.**
[14] **The mobility behavior defines the UEs movement in the simulated geography, which moving algorithm to use.**
[15] **The cell IDs tells 3Gsim which cells the UE can move through during simulation.**

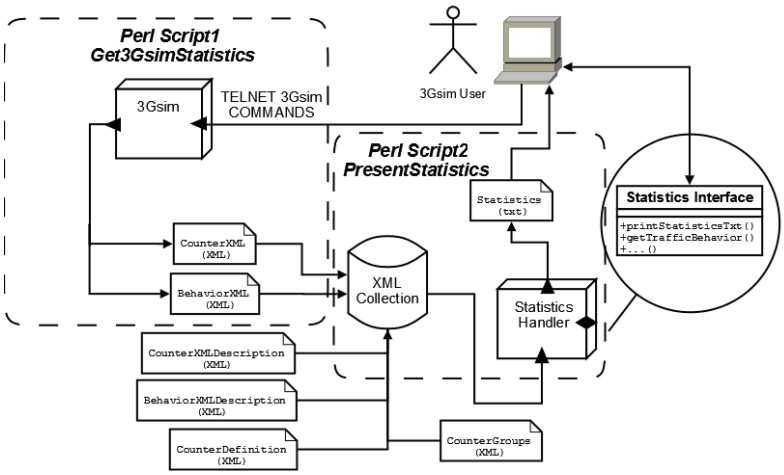

Figure 3: Statistics Handler Solution

4.1.1 Working with the Statistics Handler

As mentioned in chapter 4.1, the statistics is gathered and presented by running two Perl scripts with several flags to specify the statistics output.
The most common updates and changes the user wants to do are to add new counters to the statistics, as well as grouping counters into counter groups to more easily identify behaviors in a particular type of information.
To simplify this, specifications for counter definitions and counter groups are separated from the rest of the implementation. In the folder *xml*, in the same folder as the Perl scripts, these definitions can be found together with description documents for the *CounterXML* and the *BehaviorXML*. The description document specifies how the Statistics Handler should interpret the statistics .XML files.
If the XML statistics should change appearance in any way, only the description documents need to be updated, meaning no recompiling need to be done.

4.1.2 Improvements

An essential part, when working with the 3Gsim statistics, is that developers and users put a lot of time in making just small changes in complex Perl scripts to get the information they want. The minor and most common updates have been focused on with the new Statistics Handler, and in most cases even skip recompiling between updates.
To add or update counters or counter groups, there is no need for the developer to look in to the Java code to make changes. In the folder *xml*, the file *counterDefinition.xml* holds information about every counter recognized by the system.
A counter element looks like this:

```
<counter>
        <id>10</id>
        <dom>PS</dom>
        <p>RLC</p>
        <si>Received Packets</si>
</counter>
```

Each counter is specified by four fields. The *id* field specifies a unique counter id so that the counter easily can be added to counter groups. The *dom* field stands for domain and specifies the counter domain. At this moment the counter could either be in the packet switched[16] domain, circuit switched[17], or in an unsigned domain.
The *p* field stands for protocol and specifies the protocol used be the current counter.
The *si* field stands for Statistics Id, and could easier be called the counter name. This is not unique, the same counter name *Received Packets* could also exist in for example the circuit switched domain.
Counters defined in *counterDefinition.xml* are known to the system, and can then also be added into counter groups for a more structured view of the information. Counters unknown to the system can thus also be identified in the output document, if the user chooses to display them.

Counters that keep track of the same kind of information, benefit to error detection by being printed in the same location. Counter groups are used for exactly this and in *counterGroups.xml* a counter group is specified like in *Figure 4: Counter Group Example*.

```
<group>
        <groupId>PS Throughput Statistics</groupId>
        <counterIds>
                <cid>9</cid>
                <cid>10</cid>
                <cid>11</cid>
                <cid>12</cid>
                <cid>(11/12)</cid>
                <cid>14</cid>
                <cid>(v42*(14/12))</cid>
                <cid>17</cid>
                <cid>16</cid>
        </counterIds>
</group>
```

Figure 4: Counter Group Example

Each counter group has a *group id*, i.e. counter group name defined by the *groupId* field.
The *cid* field holds counter ids, and will print out counter id information in the same order as they are specified in this document. Also the group itself will be printed in the same order.
If the *cid* value is an integer, the statistics handler will interpret it as a counter id and print the corresponding counter name specified in counterDefinition.xml.
If the *cid* value starts with a parenthesis, the value is interpreted as math expression, dividing or multiplying counter values from two counters. In Figure 4, the fifth *cid* field has the value (11/12). This means counter values for this column will be derived from values from counter 11 divided by values from counter 12. This can be interesting when for example studying packet loss, dividing received bytes by sent bytes.
Also, if an integer in a math expression starts with the letter v, the integer is read as a value instead of a counter id. The seventh *cid* field has the value (v42*(14/12)) meaning that counter

[16] Packet switching is a network communications method that groups all transmitted data, irrespective of content, type, or structure into suitably-sized blocks or packets.
[17] A circuit switching network is one that establishes a circuit (or channel) between nodes and terminals before the users may communicate.

values for this column will be derived from values from counter 14 divided by values from counter 12 multiplied by the value 42.

4.2 Code Structure

This section explains the Statistics Handler code structure; what the different packages are responsible for.

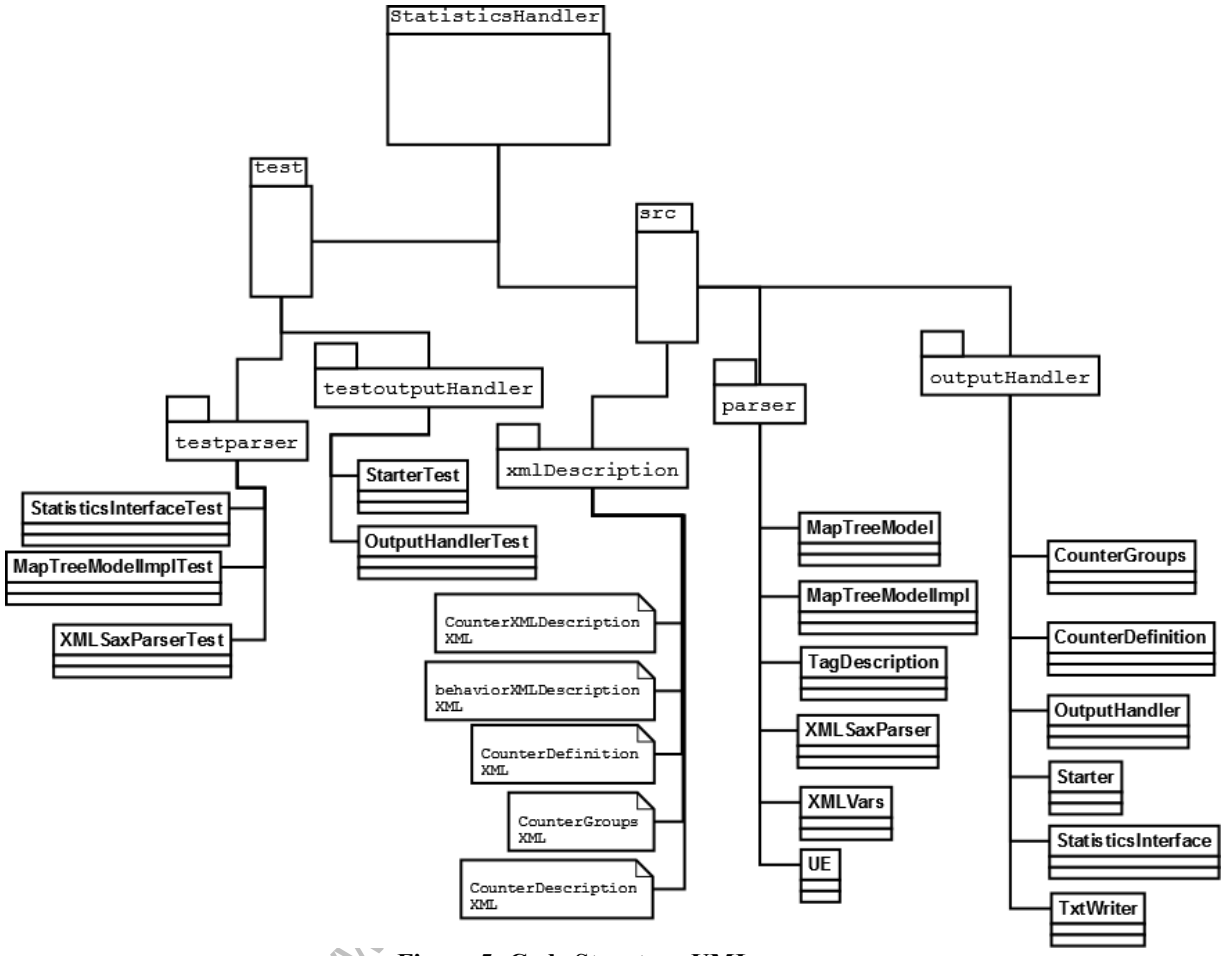

Figure 5: Code Structure UML

4.2.1 OutputHandler Package

The outputHandler package decides how information is presented, both to the user and in output files. In *CounterDefinition.java*, counter– and group definitions are gathered from *counterDefinitions.xml* and *counterGroups.xml*.

CounterDefinitions.java maps each counter group name with a number of counter IDs. The ids' counter names and values will be printed out when the group are printed in the output .txt-file. The output handler gathers information from within the Java suite and performs calculations on the data. It gathers data for the Statistics Interface, and provides the needed methods.

StatisticsInterface.java provides a limited selection of methods from *OutputHandler.java* that lets the user access the structured data and design new types of output. The OutputHandler also tells *TxtWriter.java* which rows to print.

TxtWriter writes data to the output .txt-file and as directed by the OutputHandler. It writes one line at a time.

4.2.2 Parser Package

The parser package parses information from the 3Gsim statistics XML files and handles the information according to description files in the *XMLDescription* package.
XMLSaxParser.java parses information from data according to the XML file directions stored in XML description files in package *XMLDescription*.
TagDescription.java is used by the *XMLSaxParser* to validate the parsed *tags*[18], how the tags and data parsed from within the tags should be handled. Instructions are identified and sent to MapTreeModelImpl to be executed.
MapTreeModelImpl.java takes instructions and data from *TagDescription* and creates the structured Java Suite, holding all statistics. It creates a *MapTreeModel* which simply maps different Java elements to *ArrayLists* with other *Map* objects. When all data are mapped and structured, the Map object is used as a argument for *MapTreeModel*, and thereby enables the user to run some very useful commands on the data, such as *getChildCount()*, *getRoot()*, *isLeaf()*, etc.
XMLDef.java holds definitions of all elements being created in the Java Suite. This is to simplify changes and additions to the data by the user/developer.
UE.java, defines a UE within the Statistics Handler. The most demanding calculations for the Statistics Handler comes when, for example, trying to find all counter values for UEs, that run a specific traffic behavior. Loops within loops slow down execution time quite much and it's very important not to do it more times than necessary. Gathering all such data once in an own class improves performance considerably.

4.2.3 XmlDescription Package

The XmlDescription package holds information that describes how parsed information should be handled. *counterXMLDescription.xml* describes the XML-statistics holding counter definitions and data. *behaviorXMLDescription.xml* describes the XML-statistics holding behavior and UE definitions. *counterDefinition.xml* defines all counters with a domain, protocol, name and an id. As the name implies *counterGroups.xml* defines all groups printed by the *printStatistics()* method defined in the *outputHandler/OutputHandler.printStatistics()*.

4.3 3Gsim Statistics Format

Below in *Figure 6: 3Gsim Statistics Format Description* it is illustrated how statistics information is stored within the Statistics Handler. Parent elements are Java *String* values mapped to an ArrayList of child elements of various types. To make searching more optimal, each parent name is a search path to that element. So for example, to get all data Sets, the query to the mapTreeModel would be *elementMap.get("mdc|md")*. This would return an ArrayList of data set ids. To get one of these data sets, the query would be *elementMap.get("mdc|md|ds3")*.
The '|' sign has been selected as an identifier between elements. All of these elements are specified in *parser/XMLDef.java*.

[18] ***A XML tag is the element that encapsulates values in a XML document. Example <tag> value </tag>.***

The UE class, shortly described in 4.2.2, is different than the other leaves in the Java Statistics Format. The Java UE has the characteristics of a 3Gsim UE, a unique *IMSI*, a *traffic behavior*, *mobility behavior* and a number of *cell ids*. Also, if the UE is not working properly, the UE has some status information. *isHanging*, tells if the UE behaves faulty, and the attributes *hangingRabState*, *hangingRabTime* and *hangingDeltaTime* helps explain why the UE behaves abnormally.
The UE also has all counters utilized by the UE and its corresponding counter value contained in a Java Map object.

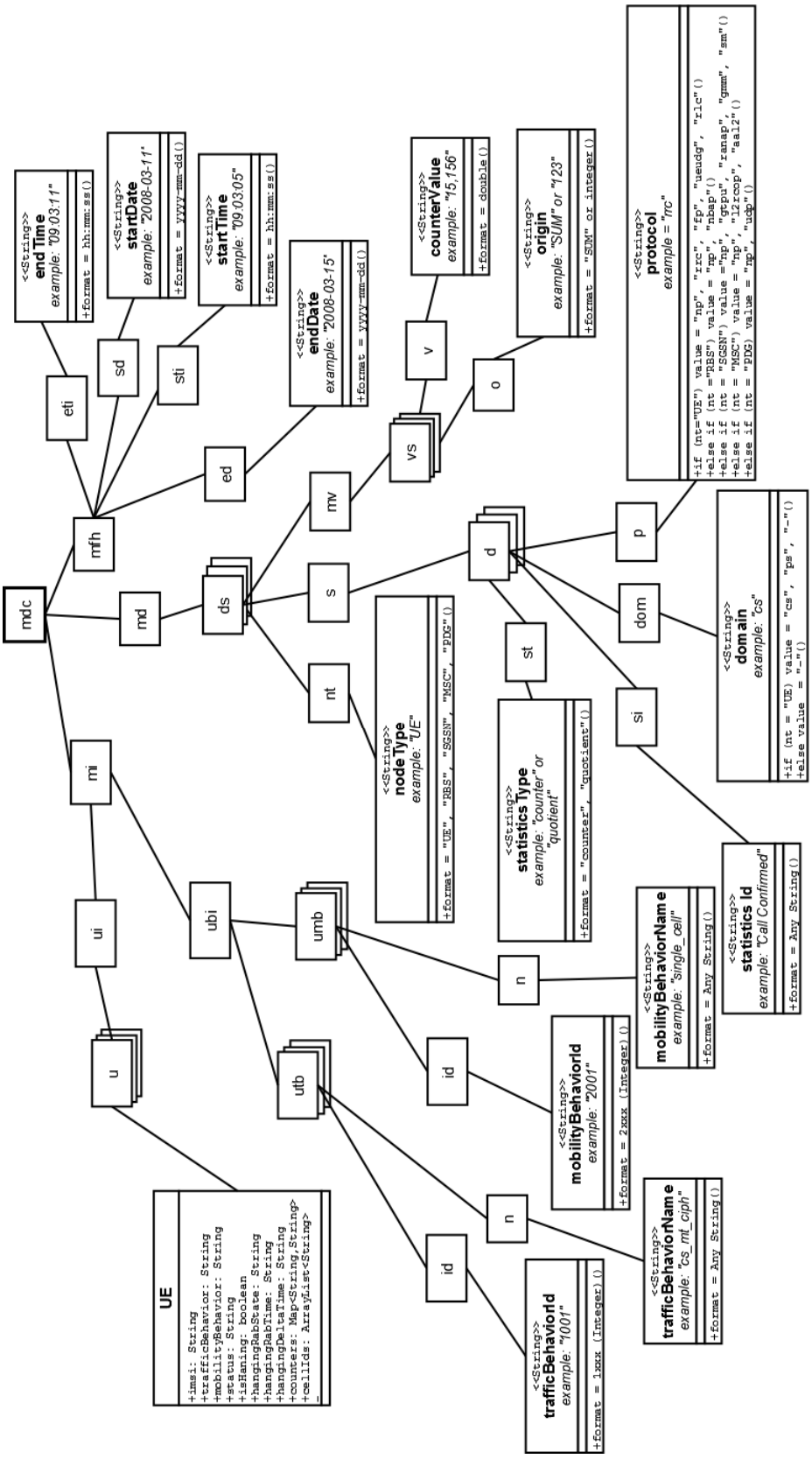

Figure 6: 3Gsim Statistics Format Description

4.3.1 Statistics Format Description

This section describes the different ids described in *Figure 6: 3Gsim Statistics Format Description.*

ui	***User Equipment Info***	***Encapsulate UEs***
u	***User (equipment)***	***Encapsulates a specification for each UE***
c	***Cell (ID)***	***ID for a specific cell***
i	***IMSI***	***The IMSI, unique for each UE***
t	***Traffic (behavior)***	***Traffic behavior ID, matches a traffic behavior listed in ueBehaviorInfo***
ubi	***UE Behavior Info***	***Encapsulate descriptions for all behaviors used in this simulation***
utb	***User (equipment) Traffic Behavior***	***Links a traffic behavior with an ID***
umb	***User (equipment) Mobility Behavior***	***Links a mobility behavior with an ID***
id	***ID***	***The unique ID for either a mobility- or traffic behavior***
nt	***Name***	***The name for either a mobility- or traffic behavior***
mfh	***Measurement File Header***	***Holds meta data for the current simulation***
eti	***End Time***	***The stop time of the current simulation***
sti	***Start Time***	***The start time of the current simulation***
ed	***End Date***	***The stop date of the current simulation***
sd	***Start Date***	***The start date of the current simulation***
md	***Measurement Data***	***Encapsulate counter/quotient data***
ds	***Data Set***	***Holds all data for a specific node Type***
nt	***Node Type***	***Specifies Node Type***
s	***Sequence***	***Encapsulate counters/quotients***
d	***Data***	***Holds data specifying a specific counter/quotient***
si	***Statistics ID***	***Specifies a counter/quotient name***
st	***Statistics Type***	***Specifies the type for a counter/quotient***
dom	***Domain***	***Describes the working domain for the counter***
p	***Protocol***	***Describes the protocol ran by the current counter/quotient***
mv	***Measurement Values***	***Encapsulate counter/quotient values***
vs	***Value Set***	***Links a counter values with a origin ID***
v	***Values***	***Counter values for a specific origin***
o	***Origin***	***Specifies a the origin of the values in this Value Set***
hs	***Hanging rab State***	***If an UE is behaving faulty thus hanging, the hs contains its' current behavior.***
hti	***Hanging Rab Time***	***The time when the UE entered the hs***
hd	***Hanging Delta Time***	***How long the UE was in the hs.***

Figure 7: 3Gsim Statistics Format Definition

4.4 Statistics Interface

Figure 8: Statistics Interface, is a description of the methods available by the Statistics interface. Each method is described by: 'method name (parameter: parameter type): return type'. After the definition comes a short description of the method.

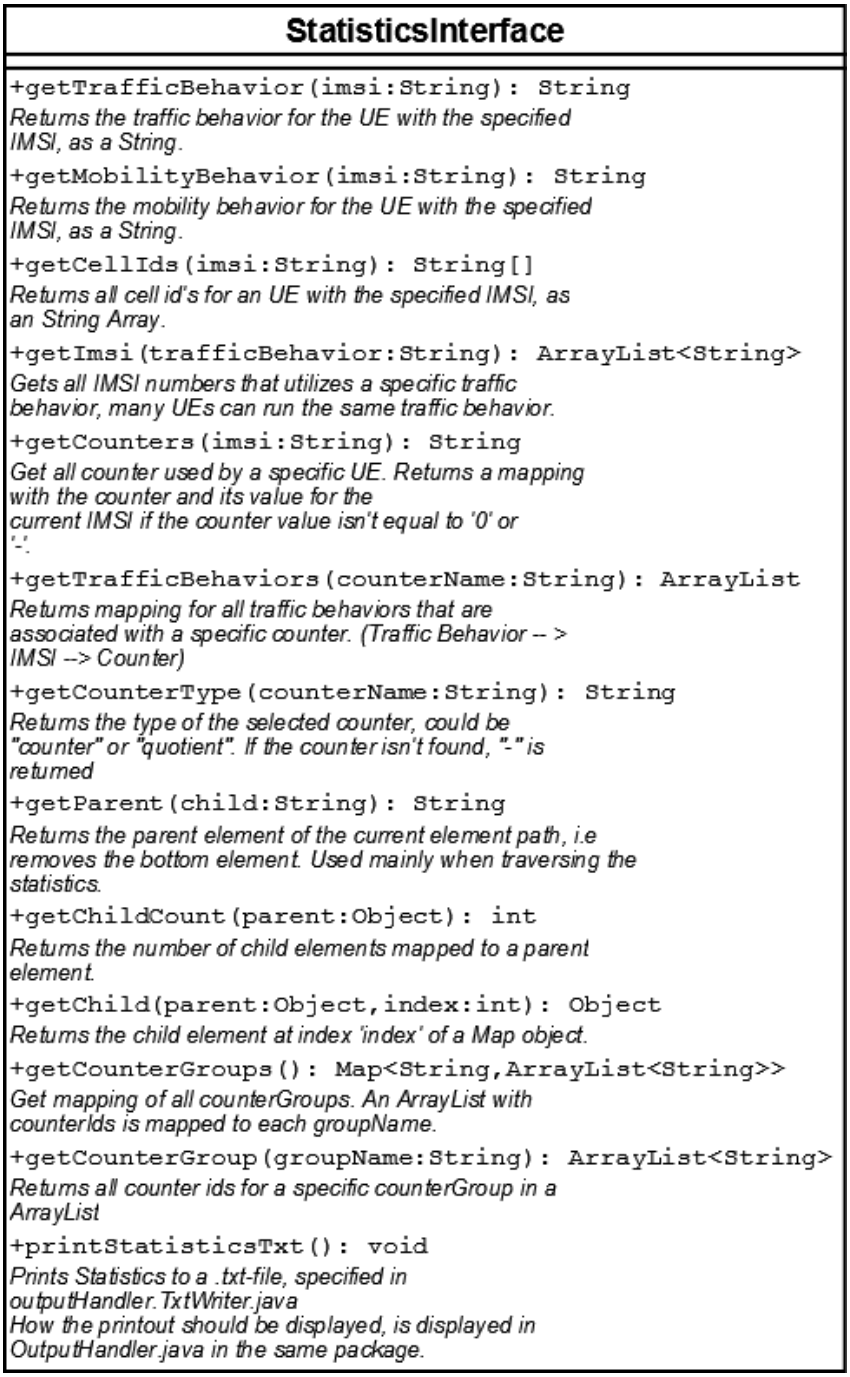

Figure 8: Statistics Interface

4.5 The Output Format

The Statistics Handler can be used to output various data using the Statistics Interface. But as mentioned in section 1.7 at least one way of viewing the data in a structured manner should be produced during this thesis work. In section 0 the conditions and limitations for this output are specified.

Data which validates similar areas of 3Gsim are clustered into groups. For example counters holding 'Packet Switched Throughput' information comes in a separate group.

PS Throughput Statistics

0.Received Bytes
1.Received Packets
2.Received Retransmissions
3.Sent Bytes
4.Sent Packets
5.Sent Retransmissions
6.Sent Retransmissions/Sent Bytes
7.Received Corrupt Packets
8.Lost Packets

Behavior	0	1	2	3	4	5	6	7	8
trafficbehavior1	1384060	8359	0	165850	3317	0	0	0	0
trafficbehavior2	480400	834	0	81300	813	0	0	0	0
trafficbehavior3	6408155	5914	1	3291129	4637	56	7.84000000	0	0
trafficbehavior4	60400	302	0	1100	11	0	0	0	0
trafficbehavior5	418800	698	0	68500	685	0	0	0	0
trafficbehavior6	632859	1293	0	111320	506	8	0.00490510	0	0
trafficbehavior7	241570	3451	0	241570	3451	0	0	0	0
trafficbehavior8	5740307	4995	0	3095586	3926	40	0.01012963	0	0
trafficbehavior9	2780780	3022	0	223720	3196	6	2.15366E-4	0	0
trafficbehavior10	528363	6790	0	352950	7059	18	0.00337922	0	0
trafficbehavior11	40600	812	0	39500	790	27	0.02870886	0	0
trafficbehavior12	2008476	2511	0	154700	3094	12	0.45818181	0	0
trafficbehavior13	752253	1524	0	336420	4806	14	0.00723247	0	0
trafficbehavior14	1000	2	0	300	3	0	0	0	0
trafficbehavior15	274958	3900	1	244288	3607	0	0	0	0
trafficbehavior16	2920616	3194	0	1191066	5478	78	0.00973782	0	0
trafficbehavior17	1222200	1469	0	141300	179	0	0	0	0
trafficbehavior18	1396108	2071	0	1170096	5514	58	0.02188286	0	0
trafficbehavior19	142000	284	0	41000	82	5	0.00148619	0	0
trafficbehavior20	1382192	15323	0	498334	8193	4	1.41050E-4	0	0
trafficbehavior21	153720	2196	0	153720	2196	0	0	0	0
trafficbehavior22	4.897056E7	34131	0	4724596	20782	42	3.73365E-4	0	0
trafficbehavior23	188860	2698	0	188860	2698	0	0	0	0
trafficbehavior24	2591266	3243	1	1207790	1364	1	8.42808E-5	0	0
trafficbehavior25	314100	221	0	10900	218	0	0	0	0
Total:	8.103460E7	109237	3	1.773589E7	86605	369	0.33545832		

Figure 9: Counter Group Output Example[19]

Figure 9: Counter Group Output Example displays a counter group printout from the .txt-file. This particular group is called *PS Throughput Statistics*. The counters are specified after the group name (counters 0-5) and for this group, all counters hold packet switched throughput related information.
After the counter names follows a table with traffic behaviors and corresponding values.
During simulation, each UE is running a specific traffic behavior, which determines how the UE operates within 3Gsim.
The values are divided into columns; one column for each counter. The values are a sum of counter values for all UEs running the specified traffic behavior.
In some cases the counters are actually mathematical expressions with two counters as variables. Consider counter number 6, *Sent Retransmission / Sent Bytes*. Values specified in this

[19] ***Because of secrecy reasons, the real traffic behavior names cannot be printed.***

column are summations of counter values from UEs running the specified traffic behavior, divided by the number of UEs. The *Total* value in the bottom of the table is in this case an average of the column values.
The Statistics Handler should also be able to handle other kind of values. The enumerator type could be a quotient. Values from 3Gsim are then presented as one value divided by another, for example '*2/2*'. These values are handled and summed as "normal" counter values except that they are always presented as one value divided by another.

Counters not known to the statistics handler can also be viewed in the output text file (how-to is specified in section 4.7)
If this option is set, two new groups, "*Non-defined Counters"* and *"Non-Group Counters"* are displayed in the end of the output document. As displayed in *Figure 10: Non-Defined Counters* their counter values are displayed in the same manner as other counter information.

```
=====================================================================
Non-Defined Counters
=====================================================================
0.UE___Successful UE GMM Attach
1.UE___Emergency CM Service Accept
2.UE___Emergency CM Service Reject
3.UE___Emergency Disconnect
4.UE___Emergency Release
5.UE___Emergency Release Complete
6.UE__FP_Received hs frame
7.UE__RRC_Handover at call setup
8.UE__RRC_Handover at call setup (including IFHO)
9.UE__RRC_Paging Type 2
10.UE__RRC_Physical Channel Reconfiguration Failures
11.UE__RRC_RAB state change
12.UE__RRC_RRC Status
13.UE__RRC_Radio Bearer Reconfiguration Failures
14.UE__RRC_Radio Bearer Release Failures
15.UE__RRC_Radio Bearer Setup Failures
16.UE__RRC_Serving E-DCH Cell Change Successes
17.UE__RRC_Transport Format Combination Controls
18.UE__RRC_URA Update Confirms
19.UE__TB_ps_cellupdate_got_cch
20.UE_PS__Successful PDP1 Activation
21.UE_PS__Successful PDP1 Deactivation
22.UE_PS__Successful Secondary PDP Deactivation
23.UE_PS__Unsuccessful PDP1 Activation
24.UE_PS__Unsuccessful PDP2 Activation
Behavior                          0          1          2          3          4          5          6
---------------------------------------------------------------------------------------------------------
trafficbehavior1                  0          0          0          0          0          0          0
trafficbehavior2                  0          0          0          0          0          0          0
trafficbehavior3                  0          0          0          0          0          0          0
trafficbehavior4                  0          0          0          0          0          0          0
trafficbehavior5                  0          0          0          0          0          0          0
trafficbehavior6                  0          0          0          0          0          0          0
trafficbehavior7                  0          0          0          0          0          0          0
trafficbehavior8                  0          0          0          0          0          0          0
trafficbehavior9                  0          0          0          0          0          0          0
trafficbehavior10                 0          0          0          0          0          0          0
trafficbehavior11                 0          0          0          0          0          0          0
```

Figure 10: Non-Defined Counters[20]

[20] ***Because of secrecy reasons, the real traffic behavior names cannot be printed.***

4.6 Testing

Automatic testing during this final thesis project was optional but a good addition according to the supervisor. Several of the departments at Ericsson are using agile methods and *TDD* (Test Driven Development) is one important part. This project work started using TDD but when time became an issue, the testing fell behind schedule and was done first after the actual application code.
As time was an issue, finalizing tests were focused on classes where further development and updates were most likely to occur.
In the parser package, *XMLParser.java* and *MapTreeModelImpl.java* test with good coverage (> 95% *statement coverage*[21] and *decision coverage*). As TDD has not been fully utilized throughout the project, some sections of the code are hard to test.
The outputHandler package, have also good test coverage for *StatisticsInterface.java*, *OutputHandler.java* and *CounterDefinition.java*.

4.7 Using the Statistics Handler

Running the Statistics Handler is very simple. While running a 3Gsim simulation the statistics can be assembled by running by a Perl script, *Get3GsimStatistics* that gathers the XML formatted statistics, and downloads it to a specified location. Different flags and arguments to the script can specify statistics output, file locations and other configurations.
To run the script locally when two valid .XML statistics documents have already been downloaded, you can instead call the Perl script *PresentStatistics*.
The script expects 2-3 arguments, the first is the path for *counterXML.xml* and the second the path for behaviorXML.xml. The last argument sets whether or not to print unspecified counters. This script is called by *Get3GsimStatistics* when xml statistics are downloaded successfully.
The Statistics Handler utilizes a number of XML documents which describes the handled information. These files are collected into the folder *xml*, and this folder needs to be in the same folder as the script for the script to run.
The output document, *Statistics.txt*, is created in the current directory.

21 ***A Practitioner's Guide to Software Test Design*** **, Copeland Lee, (2003), p.150-156**

5 Discussions and Conclusions

This chapter discusses some of the design decisions that have been made during this final thesis.

5.1 External Configuration Files

As described in section 4.1.1, XML documents, describing the structure and presentation of the output statistics, have been placed in a separate library away from the rest of the implementation. This means that updates and changes to the produced statistics can be done very easy.
CounterDefinition.xml and *CounterGroups.xml* are configuration files used to specify the output. One idea during the implementation was to just have one document, *CounterGroups.xml*, and have full definitions of all counters directly in the counter groups.
This might have simplified development further by just having one file to make updates to. Limiting the implementation to one file comes with some cost though. As all information from the configuration files are parsed, restrictions on values contained within the tags must be far more detailed when dealing with counter operations (described in section 4.1.2). Parsing (11/12) is far less susceptive to reading corrupt data than parsing for example

```
(AB state change: Speech + Int 64/64 -> * / RAB state change: * -> 2xInt EUL(10ms)/HS)
```

When parsing the string above, by for example using regexp and regular expressions, you look for patterns in the string to identify components. It is easier to identify the components in (*integer/integer*) compared to (*any combination of characters / any combination of characters*).
As any character may exist in the counter names, it is easy for the parser to get the wrong elements. Counters are constantly added to 3Gsim, and even if the Statistics Handler can process the information correct today, you should look forward and anticipate possible errors. Also, if counters should exist several times in different groups, it is easy to make mistakes when making updates and just change some of the counters.
The system also gets more resilient to misprinted counter names and attributes, thus less susceptive to human errors.

5.2 Future Improvements

During this final thesis the following thoughts on how the statistics handling could be improved were identified.

5.2.1 Identify Abnormal System Behavior

For the user to more quickly and correctly identify abnormal system behavior, it would be favorable for the system to get a configuration file of some sort as an argument. The configuration could for different counters in the system specify some average values with upper and lower limitations that automatically validate the counters current value. If the values were too far off the average value, an indication could be added to the statistics, indicating that the value was abnormal.

5.2.2 Statistics from Other Node Types

At the moment, the Statistics Handler only manages UE-counters and behaviors. But the system has been implemented that way so counters for other node types such as RBS, SGSN, MSC can just be added to the statistics without further implementation.
As described in section 4.1.2 it is still necessary to add new counters to CounterDefinition.xml and for a more structured output, to CounterGroups.xml.

5.2.3 Statistics Interface

With the Statistics Interface it is possible for the user to access statistical data in other ways than through the output .txt-file. In the output file, counter values are summed up for all counters of a specific type, it is not for example possible for the user to get statistics for a specific UE. Through the Statistics Interface, it is possible to get information for a specific UE, get all counters that utilize a particular protocol or domain or get info for just "hanging" UEs, meaning UEs that have become inactive during execution.

6 Definitions

- ***EclEmma: EclEmma is a free Java code coverage tool for Eclipse, available under the Eclipse Public License***
- ***Iub: This interface is located between the RNC (Radio Network Controller) and the RBS node.[22]***
- ***Iu-CS: This is the interface in UMTS which links the RNC with a 3G MSC. [9]***
- ***Iu-PS: This is the interface in UMTS which links the RNC with a 3G SGSN. [9]***
- ***Java: Java is a high-level object oriented programming language originally developed by Sun Microsystems and released in 1995.[23]***
- ***jUnit: JUnit is a simple framework to write repeatable tests. It is an instance of the xUnit architecture for unit testing frameworks.***
- ***Map: A Java Object that maps keys to values.[24]***
- ***MapTreeModel: The Java MapTreeModel takes a Java Map object as a parameter and creates a traversable tree structure. [25]***
- ***Perl: Perl is a high-level programming language with an eclectic heritage written by Larry Wall.[26]***
- ***Telnet: TCP/IP based application protocol that enables a terminal to interact with a program running in another computer:***
- ***Xerces: Xerces is a library for parsing, validating and manipulating XML documents.[27]***
- ***3G: 3G is the third generation of tele standards and technology for mobile networking, superseding 2.5G.[28]***
- ***3Gsim: 3Gsim is a load generator for traffic simulation in a WCDMA network, for verification of the RNC and the RAN***

22 ***Mpirical companion*** **(2002) [www] , Mpirical limited, http://www.mpirical.com/companion/mpirical_companion.html, 2009-01-06**

23 ***The Source for Java Developers*** **(1994) [www], Sun Microsystems, Inc., http://java.sun.com/, Retrieved 2008-09-23**

24 ***Interface Map*** **(2008) [www], Sun Microsystems, Inc., http://java.sun.com/javase/6/docs/api/java/util/Map.html, Retrieved 2008-09-30**

25 ***MapTreeModel*** **(2001) [www], Christian Kaufhold http://www.chka.de/swing/tree/MapTreeModel.html, Retrieved 2008-09-26**

26 ***What is Perl*****? (1997) [www], Cristiansen Tom, Nathan Tokington, http://perldoc.perl.org/perlfaq1.html, Retrieved 2009-02-18**

27 ***The Apache Project*** **(1999) [www], Apache Software Foundation http://xerces.apache.org/, Retrieved 2008-09-26**

28 ***About mobile technology and IMT-2000*** **(2005) [www], http://www.itu.int/osg/spu/imt-2000/technology.html#Cellular%20Standards%20for%20the%20Third%20Generation, Retrieved 2009-01-15**

References

- *Arabloui Sona, Supporting problem detection in a network traffic simulation system (2008), Department of Computer and Information Science, Linköping*
- *Apache Software Foundation, The Apache Project (1999) [www], http://xml.apache.org/, 29th of September 2008*
- *Cristiansen Tom, Nathan Tokington, What is Perl? (1997) [www], http://perldoc.perl.org/perlfaq1.html, Retrieved 18th of February 2009*
- *Copeland, Lee, A Practitioner's Guide to Software Test Design (2003) Artech House, Incorporated*
- *Ericsson AB, Ericsson in Brief [www], http://www.ericsson.com/ericsson/corpinfo/index.shtml, Retrieved 6th of January 2009.*
- *FileFormat Info, Parsing with regexp [www], http://www.fileformat.info/tool/regex.htm, 24th of September 2008*
- *Goyvaerts Jan, Regular Expressions (2008) [www], http://www.regularexpressions.info/, 24th of September 2008*
- *Hoffmann Marc R, Eclemma (2001) [www], www.eclemma.org, Retrieved 10th of October 2008*
- *International Telecommunication Union, About mobile technology and IMT-2000 (2005) [www], http://www.itu.int/osg/spu/imt-2000/technology.html#Cellular%20Standards%20for%20the%20Third%20Generation, Retrieved 2009-01-15*
- *McLaughlin Brett, Java & XML(2000) [www], http://java.sun.com/developer/Books/xmljava/ch03.pdf, Retrieved 24th of September 2008*
- *Mpirical limited, Mpirical companion (2002) [www], http://www.mpirical.com/companion/mpirical_companion.html, 2009-01-06*
- *Lundqvist Malin, Merkel Magnus, Önnegren Britta. Lathund för rapportskrivning (2006) [www]. http://www.liu.se/content/1/c6/11/00/75/Lathund.pdf. Retrieved 10th of October 2008*
- *Object Mentor(www.objectmentor.com), jUnit (2000) [www], http://www.junit.org/, Retrieved 10th of October 2008*
- *Sun Microsystems, Inc., The Source for Java Developers (1994) [www], http://java.sun.com/, Retrieved 23rd of September 2008*
- *Sun Microsystems Inc., Interface Map (2008) [www], http://java.sun.com/javase/6/docs/api/java/util/Map.html, Retrieved 2008-09-30*

På svenska

Detta dokument hålls tillgängligt på Internet – eller dess framtida ersättare – under en längre tid från publiceringsdatum under förutsättning att inga extraordinära omständigheter uppstår.

Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för ickekommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns det lösningar av teknisk och administrativ art.

Upphovsmannens ideella rätt innefattar rätt att bli nämnd som upphovsman i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart.

För ytterligare information om Linköping University Electronic Press se förlagets hemsida *http://www.ep.liu.se/*

In English

The publishers will keep this document online on the Internet - or its possible replacement - for a considerable time from the date of publication barring exceptional circumstances.

The online availability of the document implies a permanent permission for anyone to read, to download, to print out single copies for your own use and to use it unchanged for any non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional on the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its WWW home page: *http://www.ep.liu.se/*