Juho Pohjala

Tool for network level configuration and auditing in mobile backbone network

School of Electrical Engineering

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo 16.5.2011

Thesis supervisor:

Docent Kalevi Kilkki

Thesis instructor:

M.A. Jukka Malinen

Aalto University
School of Electrical
Engineering

AALTO UNIVERSITY
SCHOOL OF ELECTRICAL ENGINEERING

ABSTRACT OF THE
MASTER'S THESIS

Author: Juho Pohjala

Title: Tool for network level configuration and auditing in mobile backbone network

Date: 16.5.2011 Language: English Number of pages: 8+55

Department of Communications and Networking

Professorship: Network Economics Code: S-38

Supervisor: Docent Kalevi Kilkki

Instructor: M.A. Jukka Malinen

The purpose of this thesis is to implement a prototype for the configuration and auditing of network elements in the mobile core network. The focus of the prototype is on the IP level. The prototype is needed in order to evaluate the feasibility of this new type of tool. The feedback from the prototype evaluation provides input for the full-scale implementation of the network level configuration and auditing tool.

The Waterfall model was used to steer the prototype development, because the model was considered suitable for small-scale software development projects. The user interface of the prototype is based on the spreadsheet design, which was considered suitable for displaying network structures. Finally, Java was chosen as the programming language because of its platform independence.

The prototype development was a success, because the prototype is capable of performing network level auditing and relatively simple configurations. The prototype was considered as a possible choice for a network level configuration and auditing tool. In addition, it was thought that the full-scale version of the tool could simplify and speed up network management tasks.

From a technical perspective, no problems were found during the prototype development that would prevent the full-scale implementation of the tool. This thesis provides some suggestions for the full-scale development. For example, it is recommended to analyse whether the user interface of the tool should be renewed.

Keywords: network management, core network, Waterfall model, user interface, IP

AALTO-YLIOPISTO
SÄHKÖTEKNIIKAN KORKEAKOULU

DIPLOMITYÖN
TIIVISTELMÄ

Tekijä: Juho Pohjala

Työn nimi: Työkalu verkkotason konfigurointiin ja auditointiin matkapuhelinrunkoverkoissa

Päivämäärä: 16.5.2011 Kieli: Englanti Sivumäärä: 8+55

Tietoliikenne- ja tietoverkkotekniikan laitos

Professuuri: Tietoverkkotalous Koodi: S-38

Valvoja: Dos. Kalevi Kilkki

Ohjaaja: FM Jukka Malinen

Tämän työn tarkoituksena on toteuttaa prototyyppi, jolla voi hallita ja valvoa runkoverkon verkkoelementtejä IP-tasolla. Prototyyppiä tarvitaan uudentyyppisen ohjelmiston soveltuvuuden arviointiin, josta saatua tietoa voidaan myöhemmin hyödyntää kokoversion toteutuksen yhteydessä.

Prototyyppi kehitettiin vesiputousmallin avulla, joka todettiin sopivaksi pienimuotoisen ohjelmiston kehitykseen. Prototyyppi perustuu taulukkolaskentaohjelmista tuttuun käyttöliittymään, jonka arvioitiin sopivan hyvin verkon struktuurin kuvaamiseen. Ohjelmointikieleksi valittiin Java, koska sen avulla voitiin taata alustariippumattomuus.

Prototyypin kehitys onnistui hyvin, sillä prototyypillä on mahdollista auditoida verkkoja sekä tehdä suhteellisen yksinkertaisia konfigurointeja. Prototyyppi todettiin yhdeksi vaihtoehdoksi toteuttaa työkalu verkkotason hallinnointiin ja valvontaan. Lisäksi työkalun kokoversiolla arvioitiin olevan mahdollista yksinkertaistaa ja nopeuttaa verkkojen hallintaa.

Työkalun kehityksen aikana ei löydetty teknisiä esteitä sen jatkokehitykselle. Kokoversion toteutuksen yhteydessä kannattaa hyödyntää tässä työssä esitettyjä suosituksia. On esimerkiksi suositeltavaa tutkia, pitääkö työkalun käyttöliittymään tehdä muutoksia.

Avainsanat: verkonhallinta, runkoverkko, vesiputousmalli, käyttöliittymä, IP

Preface

This thesis was written at Oy LM Ericsson Ab in Finland. I am grateful for the opportunity to write my final thesis here.

First, I want to thank Tomas Nordman and Christer Hamberg for your guidance regarding the tool development and also for your technical insight that you shared with me. Thanks to Tomas for making the tool development project and this thesis possible. Second, I want to thank my supervisor Kalevi Kilkki and instructor Jukka Malinen for your comments, advice and efforts during the thesis work. Thanks also to William Martin for proofreading the final version of this thesis.

Last, but not the least, I want to thank my wife Nannan for your continuous support and patience while I've been working on this thesis.

Jorvas, Kirkkonummi, 16.5.2011

Juho Pohjala

Contents

Abbreviations

2G	Second Generation
3G	Third Generation
3GPP	3rd Generation Partnership Project
API	Application Programming Interface
ATM	Asynchronous Transfer Mode
AuC	Authentication Center
AWT	Abstract Window Toolkit
BSC	Base Station Controller
BTS	Base Transceiver Station
CAPEX	CAPital EXpenditure
CCR	Customer Configuration Requirement
CN	Core Network
CPU	Central Processing Unit
CS	Circuit Switched
DoS	Denial of Service
EIR	Equipment Identity Register
GCP	Gateway Control Protocol
GMP	Generic M-MGw Package
GSM	Global System for Mobile communications
GUI	Graphical User Interface
HLR	Home Location Register
HW	Hardware
IEEE	Institute of Electrical and Electronics Engineers
IMEI	International Mobile Equipment Identity
IP	Internet Protocol
ITU	International Telecommunication Union
ITU-T	ITU's Telecommunication Standardization Sector
JRE	Java Runtime Environment
JVM	Java Virtual Machine
LTE	Long Term Evolution
M-MGw	Mobile Media Gateway
MML	Man-Machine Language
MO	Managed Object
MOM	Managed Object Model
MSC	Mobile Switching Center
MSC-S	Mobile Switching Center Server
MSS	Ericsson Mobile Softswitch Solution
O&M	Operation and Maintenance
OPEX	OPerational EXpenditure
OS	Operating System
OSS-RC	Operations Support System, Radio and Core network
PLMN	Public Land Mobile Network
PS	Packet Switched

PSTN	Public Switched Telephone Network
QoS	Quality of Service
RAN	Radio Access Network
RBS	Radio Base Station
RNC	Radio Network Controller
SAE	System Architecture Evolution
SRS	Software Requirements Specification
SSH	Secure SHell
SW	Software
TCP	Transmission Control Protocol
TDM	Time Division Multiplexing
UE	User Equipment
UI	User Interface
UMTS	Universal Mobile Telecommunications System
VBA	Visual Basic for Applications
VLAN	Virtual Local Area Network
VLR	Visitor Location Register
VMGw	Virtual Media Gateway
WCDMA	Wideband Code Division Multiple Access
XML	eXtensible Markup Language

1 Introduction

This chapter gives a brief overview of the topic related to this thesis. At first, the background of the thesis will be introduced, describing why the topic is important. Then, the research problem and the goals are specified, followed by the scope for this thesis. Finally, the structure of the thesis will be described at the end of this chapter.

1.1 Motivation

Network configuration can be a challenging task in the present day's communication networks. Reconfiguration is needed, for instance when a network node is migrated from one network to another, or when an IP address is changed. Getting the node back up and running after reconfiguration is a top priority for both network operators and network vendors that provide support.

As modern communication networks are highly complicated, node configuration is a time-consuming task, and misconfigurations are quite common, too. Getting a good overview of the network structure is a challenging task, and that is why problems are typically on a network level rather than on a node level. Moreover, finding some small configuration error from the network might require a lot of work and effort, which, in turn, could cause delays in different node deployment projects. Nevertheless, the configuration work must be done with great precision in order to take full advantage of the network resources and to avoid misconfigurations that could potentially have a negative impact on the network performance. In addition, fast recovery from misconfigurations is essential, since it could possibly avoid a problem escalating to a larger scale.

Network auditing, or supervision, is also of high importance, since it contributes to service reliability and is essential for overall network performance. The goal is to make sure that everything works as planned, and in case that some disturbances are experienced, corrective measures can be initiated as soon as possible. In an optimal case, any problems are discovered proactively, i.e. before any major disturbances are experienced. Some examples of network level issues are a misconfigured interface and a hardware failure, which may, for instance, reduce the available network capacity. In practice, auditing a single network component might be easy, but when it comes to auditing the whole network, the task often turns out to be more difficult.

All in all, figuring out the root cause for some network level issue as well as efficient auditing of the network might be challenging tasks that could require lots of time and effort. This is the starting point for this thesis, since there is some space for improvements in terms of network configuration and auditing. Next, the research problem and the goals of this thesis are introduced in the following section.

1.2 Research problem and goals of the thesis

As of today, almost every single network element has its own configuration tool. Thus, getting an understanding and a clear network level overview is rather slow,

since configuration data must be fetched from the network using multiple tools. For this reason, configuring and auditing a network has its challenges. There is a need for a tool that could handle configuration and supervision on a network level, supporting several node types, and thus, minimising the number of required tools as well as the manual work that is currently required.

To overcome the current issues regarding network level configuration and auditing, a tool is designed as a part of this thesis. The goal is to develop a prototype for network level auditing and configuration, enabling faster, easier and more efficient task handling than previously has been possible. Such a tool should be able to read and modify network parameters as well as audit the network configuration faster than before. With proper implementation, the tool could give a quick overview of the network status and configuration, which is useful in many ways. For example, the tool could help solving customer service requests, because analysis of the network structure would be faster.

However, before fully implementing the tool, it must be ensured that the auditing and configuration functionalities can be combined together efficiently. This is the research problem for this thesis. After all, there is very little need for a tool that is not able to make task-handling any easier or faster than previously. Getting this kind of information, as early in the development phase as possible, is valuable as such.

In the end, the proposed solution is evaluated. The idea is that the tool could be used at Ericsson's test environment, in order to analyse its strengths and weaknesses. Based on the findings, the tool development could either be continued after the thesis, or discarded.

To conclude, this thesis tries to answer the following set of questions:

- Is it feasible to combine network level configuration and auditing functionalities in the same tool?
- Could the tool be used to simplify or speed up the tasks that involve network configuration and auditing?
- Can the tool provide a quick overview of the network configuration and status?

The goals of this thesis include finding answers to the questions above as well as developing a working prototype. Next, in the following section the scope of the thesis is explained in more detail.

1.3 Scope of the thesis

As stated in the previous section, one of the main goals of this thesis is to find out whether network configuration and auditing functionalities could be combined together efficiently. To answer this question, it is not necessary to include all network components in the tool implementation. In fact, it should be enough to support only a few key elements from the core network. However, the chosen elements must not be unique, i.e. a network should contain multiple instances of each node type,

otherwise configuring and auditing on the network level would not be feasible. For these reasons, and due to Ericsson Finland's involvement in Mobile Media Gateway (M-MGw) development, the tool for this thesis focuses primarily on Mobile Media Gateways. In addition, network level auditing of Mobile Switching Centers (MSCs) is included in the tool.

Furthermore, this thesis focuses only on IP level, meaning that other bearers such as Asynchronous Transfer Mode (ATM) and Time Division Multiplexing (TDM) are excluded from this thesis work. There are three main reasons for the tool to operate only on an IP level. First, at Ericsson there is a need for an IP level configuration and auditing tool. Second, the usefulness of the tool can be evaluated using just the IP level. Third, operator networks are evolving towards an all-IP architecture [1], [2]. Because of this transition, the selection of IP seems natural.

Extra attention must also be paid to tool design, since it should allow the addition of other bearers and nodes that could be introduced later on. It should be possible to extend the support without the need for total redesign. For instance, support could be added for the Radio Network Controller (RNC), Base Station Controller (BSC) as well as routers and switches. In future, the network configuration and auditing tool could also support not only IP, but also ATM and TDM.

Finally, the tool for network configuration and auditing is designed for networks that are equipped with Ericsson's network elements. Support for other network vendors is not considered in the scope of this thesis.

1.4 Structure of the thesis

This thesis is divided into seven chapters. Chapters 2–4 cover the background and theory. In Chapter 5, the tool for network level configuration and auditing is discussed. The results of the tool implementation are presented in Chapter 6, followed by discussion and conclusions in Chapter 7. The last chapter also includes suggestions for future development.

2 MSS core network architecture

In this chapter, the background of the thesis is discussed. The tool that is developed as a part of this thesis, operates in the Ericsson mobile packet backbone network, thus understanding its basic concepts is essential. First, an overview of the Ericsson Mobile Softswitch Solution (MSS) is given. The second section introduces the transition towards all-IP architecture that has been ongoing in the operator networks. Then, the key network elements of the MSS architecture are presented. These key nodes are the Mobile Media Gateway (M-MGw) and the Mobile Switching Center Server (MSC-S). At the end of this chapter, the other MSS-related network elements are discussed.

2.1 Overview of the Mobile Softswitch Solution (MSS)

The Ericsson Mobile Softswitch Solution is comprised of two main network elements, the Mobile Media Gateway (M-MGw) and Mobile Switching Center Server (MSC-S). In contrast to the classical MSC architecture, in which the MSC handles both control and switching traffic, the MSS architecture implements a layered architecture by separating control and switching into separate nodes. In the MSS, the M-MGw handles switching, while the MSC-S handles call control.

Figure 1 visualises the layered architecture of the Ericsson Mobile Softswitch Solution. As shown in the figure, the MSC-S is located in the control layer, thus being responsible for the signalling traffic in the network. The M-MGw, on the other hand, handles the actual network payload in the connectivity layer. [3]

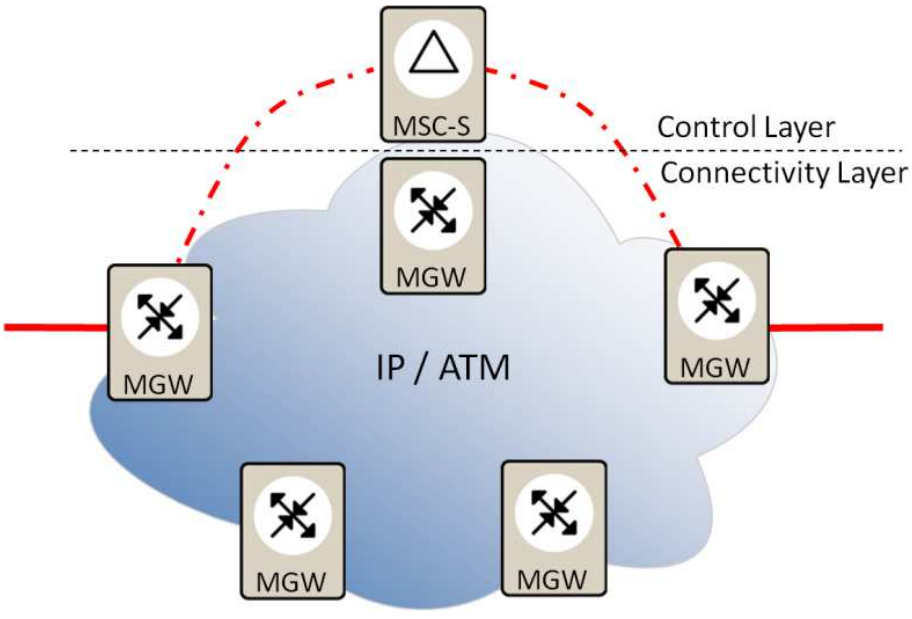

Figure 1: The layered architecture of the MSS. [3]

The physical distance between the M-MGw and MSC-S is not limited. Hence, the MSC-S can be located at a central site, taking care of the signalling and managing the connected M-MGw nodes. The M-MGw nodes, on the other hand, can be deployed at remote sites where the majority of traffic can be connected locally. In other words, a given payload does not always need to travel through the entire network, but instead, it may travel via the shortest path using the most efficient coding. This type of transmission that renders less traffic through the core of the network increases network efficiency and may decrease the operational expenditure (OPEX) of teleoperators. This is due to the fact that the majority of calls in most networks are locally terminated. [3]

In addition to the geographical flexibility, the layered architecture has another benefit. It makes it possible for the teleoperators to achieve convergence between the circuit-switched and packet-switched components of the network. This is vital for a smooth evolution towards all-IP networks. Section 2.2 (p. 6) presents more information about the ongoing transition towards all-IP network architecture.

On the other hand, the layered architecture adds more complexity to the network. Instead of only one node type, there are two node types to be managed. Hence, the importance of network planning is emphasised, and also network management might be more challenging. Next, the control and connectivity layers are briefly discussed in the following subsections.

2.1.1 The control layer

The control layer provides all the functionality needed for high-quality and seamless services across various types of networks. In general, the network elements on the control layer are responsible for the signalling traffic in the network. In other words, they handle, for instance, mobility management, set-up and release of calls, security and certain circuit-mode supplementary services. [4]

The key network element on the control layer is the Mobile Switching Center Server. This node is responsible for e.g. call control, including bearer services, charging and security. The MSC-S also controls different transport networks. Section 2.4 (p. 11) introduces the MSC-S in more detail.

An important protocol on the control layer is the GCP (Gateway Control Protocol). The MSC-S uses the GCP to control the connected M-MGw nodes. The Gateway Control Protocol is based on the ITU-T H.248.1 standard, and it is used for both transmitting and receiving information between the MSC-S and M-MGw. The GCP may use either IP or ATM as the signalling transport. Furthermore, the GCP provides various procedures for call-related functions, such as tone sending and bearer establishment, as well as call-independent functions, such as M-MGw load control. These procedures conform to the 3GPP TS 29.232 specification. [5]

2.1.2 The connectivity layer

The connectivity layer provides interfaces to different networks such as the 2G and 3G radio access networks and the public switched telephone network (PSTN). The connectivity layer provides a pure transport mechanism capable of transmitting the

actual service traffic. In other words, the connectivity layer transports voice and data services as well as various multimedia-related streams.

The backbone architecture of the connectivity layer consists of core and edge network components. The core components typically consist of routers, while the edge components consist of M-MGw nodes. The M-MGw, which is the key network element on the connectivity layer, is controlled by the MSC-S. In general, the M-MGw provides call switching by connecting calls within the same or between different networks. In addition, the M-MGw provides e.g. coding and decoding of speech streams, transport protocol conversion and echo cancelling [4]. In Section 2.3 (p. 8), the M-MGw is discussed in more detail.

As shown in Figure 1 (p. 4), the connectivity layer is based on ATM (Asynchronous Transfer Mode) or IP (Internet Protocol), i.e. it operates in the packet-switched domain [6]. This is quite different from the classical MSC architecture, which is a TDM-based (Time Division Multiplexing) monolithic core network utilising thousands of point-to-point links [3]. The packet-switched domain enables efficient speech coding, thus reducing the bandwidth needed for voice transmission.

As the connectivity layer is based on ATM or IP, it allows the teleoperators to provide various QoS (Quality of Service) mechanisms. With QoS it is possible to provide agreed service and a certain level of performance to end-users by, for example, prioritising different applications or users. For instance, QoS may be used to guarantee the minimum bit rate or the maximum delay. Without a well-designed QoS policy, there is a risk that the network functions less efficiently, causing delays in time-sensitive services. This, in turn, may have a negative impact on the customer satisfaction. QoS can also be used for generating extra revenues, because some customers may be willing to pay for a premium service with, for instance, a guaranteed bit rate [7]. In general, QoS plays a central role in modern communication networks, thus the ability to guarantee appropriate Quality of Service is an essential feature provided by the connectivity layer.

2.2 Migration towards pure IP transit architecture

During the past decade, teleoperators have been replacing their existing core networks from TDM or ATM transport technologies to pure IP transport [8]. In general, there is a clear trend in which CS (Circuit Switched) technologies are replaced by PS (Packet Switched) technologies, but the transition from ATM towards IP is also clearly visible. It could be said that the support for IP guarantees a future-proof solution for modern communication networks. Hence, IP-related competence and network management tools that operate on an IP level are getting increasingly important.

The evolution towards all-IP networks will result in better speech quality due to less transcoding, more efficient bandwidth usage as well as reduced costs [8]. It will also reduce the complexity of dimensioning the point-to-point TDM links in today's communication networks [3]. In general, the benefits of an all-IP architecture are evident. These main benefits are presented in Table 1.

Table 1: The benefits of an all-IP network. [7]

Reduced costs
Using IP in both the core network (CN) and radio access network (RAN) nodes minimises capital expenditure (CAPEX) as well as operational expenditure (OPEX). First of all, IP-based transport is cheap on a bit-per-second basis. Second, IP traffic scales well and adapts dynamically to various bit rate and bandwidth requirements. In other words, the continuously growing demand for higher bandwidth can be accommodated by the operator more easily. Third, the IP-based solution is relatively simple, which leads to reduced management and deployment costs.

Efficient use of existing infrastructure
The transition towards an all-IP architecture does not, however, take place overnight. Thus, for the operators it is crucial to maintain their ongoing operations and revenues throughout the network transition. This transition, however, can be done by utilising the existing network, at least to some extent. In practice, the migration towards all-IP could be started from the legacy infrastructure. The transport networks, previously used for TDM, could be upgraded for IP packet handling. All of this could be done when the capacity requires it, without halting the ongoing business operations. Finally, as the IP network is based on standards, the interoperability of the network infrastructure can be guaranteed.

Increased network security
IP has its own vulnerabilities that are due to e.g. increased connectivity and increased adoption of IP devices. One of these problems is the vulnerability to various denial-of-service (DoS) attacks. Nevertheless, the IP networks can be built on a set of security features that can provide the adequate level of network security. This has the additional benefit of lowering costs, because less dedicated security nodes are required, and also the need for administrative processes is reduced.

Guaranteed Quality of Service
With IP it is possible to provide telecom-grade Quality of Service (QoS). For example, traffic differentiation is possible. QoS allows grouping of different traffic types depending on the delay sensitivity. Also various queuing and scheduling algorithms are possible, as well as admission control. The goal of admission control is to ensure that the network capacity is not overloaded, and if needed, there are methods to limit the traffic that is causing congestion.

High availability
High availability can be integrated into an all-IP network with careful network design. The network must be built in a way that if an individual link

or node fails, traffic can still be routed to its destination. IP offers certain built-in mechanisms that can provide availability and redundancy. For example, link or node failures can be detected with standard routing protocols. It is also possible to move the traffic onto alternative paths, if the primary path is down. In order to ensure high availability, it is essential that the network elements are resilient. Resiliency can be achieved through e.g. hardware redundancy.

Assured migration path
The next-generation networks, like LTE (Long Term Evolution), are IP-based. An important part of the LTE is the System Architecture Evolution (SAE), which is an IP-based core network architecture of the LTE standard. As it is known that the next-generation networks operate on the IP level, it is already possible to start investing in the future. An all-IP network is the solution for preparing the way for the LTE.

Efficient network management
Basically, a simple network architecture is also simple to manage. Thus, an all-IP network allows cost-effective and integrated management.

2.3 Media Gateway for Mobile Networks (M-MGw)

The Mobile Media Gateway is a key component in the Mobile Softswitch Solution. As shown in Figure 1 (p. 4), it is located in the connectivity layer of the layered architecture. The M-MGw nodes handle the service traffic in the network, thus they are normally deployed at remote sites and at interconnection points with other mobile and fixed networks [3]. In this way it is possible to cut transmission costs, because the majority of the calls are locally terminated [3]. The M-MGw traffic handling is controlled by the MSC Server using the Gateway Control Protocol (GCP).

Figure 2 shows the interconnections and interfaces of the M-MGw. As shown in the figure, the M-MGw connects the mobile core network with the 2G and 3G networks, the public switched telephone network (PSTN) as well as other external networks. These interconnections can be based on TDM, ATM or IP [6]. In addition, the M-MGw bridges various transmission technologies and adds certain services to end-user connections. Examples of such services are echo cancellation and voice-quality enhancements [6].

Because an M-MGw node can be connected simultaneously to 2G and 3G networks, it increases network efficiency, since less intersystem signalling is generated. As a result, teleoperators may achieve cost-savings. In addition, the capacity allocation between 2G and 3G is automatic and dynamic [6].

As discussed in the previous section, support for IP is the key for a future-proof network architecture. As M-MGw supports TDM, ATM and IP interfaces, M-MGw should also support the transition towards next-generation networks. However, support for many interfaces alone is not enough, as also increasing capacity requirements

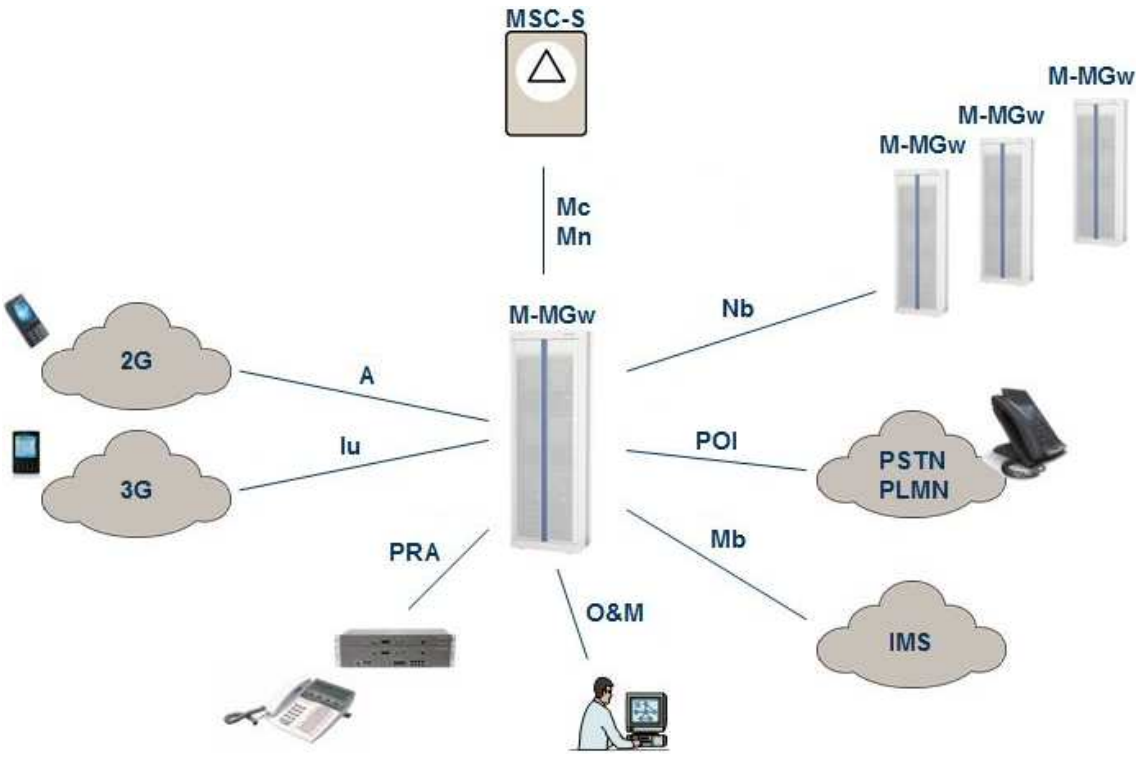

Figure 2: The interconnections and interfaces of the M-MGw. [5]

need to be satisfied.

The M-MGw supports MSC pooling through a feature known as the virtual media gateway (VMGw) [6]. The VMGw functionality allows several MSC Servers to use the resources in one physical M-MGw [5]. Furthermore, an MSC-S can choose an M-MGw close to the call's termination point in order to optimise network load [5]. The MSC pooling also provides additional network level redundancy.

2.3.1 Hardware generations and software releases

The M-MGw has seen numerous hardware generations and software releases after it was first released to market. The currently supported hardware generations are GMPv2 (Generic M-MGw Package version 2), GMPv3 and GMPv4. The latest major software release is R6 (release 6), which together with GMPv4 can offer more capacity and better performance than any earlier release.

Figure 3 presents the hardware generations that are currently supported together with the M-MGw software releases. As shown in the figure, it is possible to upgrade any hardware configuration to the latest software release. This possibility offers cost-effectiveness, since the mobile operators can get the new features that have been introduced to new releases simply with a software upgrade. In addition, new software releases may offer better performance as well as fixes to some software-related faults.

The tool that is developed as a part of this thesis, is intended for R5 and R6

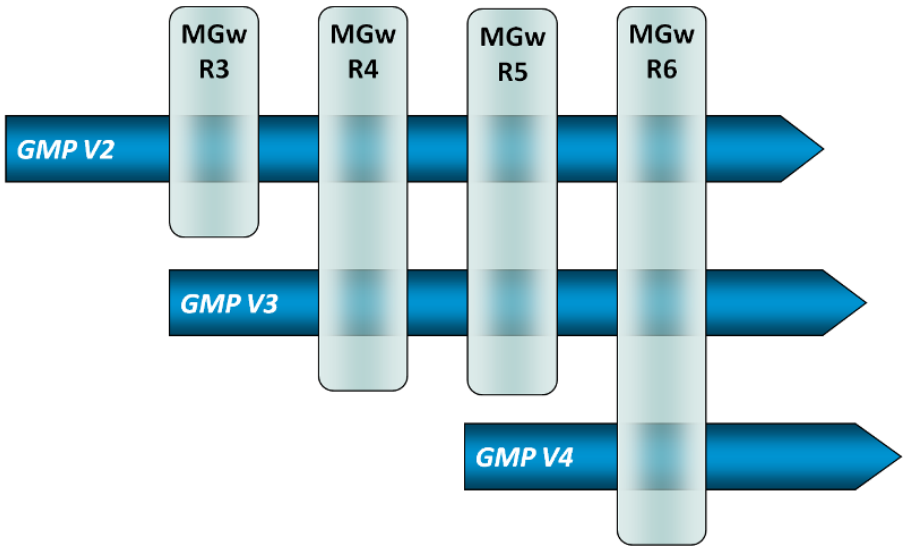

Figure 3: The relationship between M-MGw SW releases and HW generations. [5]

software releases. Thus, GMPv2, GMPv3 and GMPv4 hardware generations are supported. Next, the following subsection introduces the Managed Object Model, which is an important concept in terms of this thesis.

2.3.2 Managed Object Model (MOM)

The Managed Object Model describes the configuration of an M-MGw node. It is an important concept, because it provides all the necessary information and means for node management and configuration. The tool that is developed as a part of this thesis, is based on reading and manipulating the MOM, thus understanding the basics is essential.

The MOM consists of several Managed Objects (MOs). A Managed Object is an abstract view of a resource that is subject to management. An MO can be a representation of both physical and virtual objects. For example, a certain MO could represent some program that runs in the node, or a physical link between two interfaces. Managed Objects have a certain number of attributes, depending on the MO type. The attributes contain a value, which can be an integer, character string, or e.g. a reference to some other MO. In other words, an MO can be defined in terms of the attributes it has, operations it can perform, notifications it can issue and interactions it can have with other Managed Objects.

All Managed Objects and their attributes as well as MO classes and their relationships between other classes are defined in the Managed Object Model. MOM also contains information regarding default values, mandatory fields and value ranges of MO attributes. Recommended MO attribute values are defined in the M-MGw user guide. For clarification, the structure of MOM is shown in Figure 4.

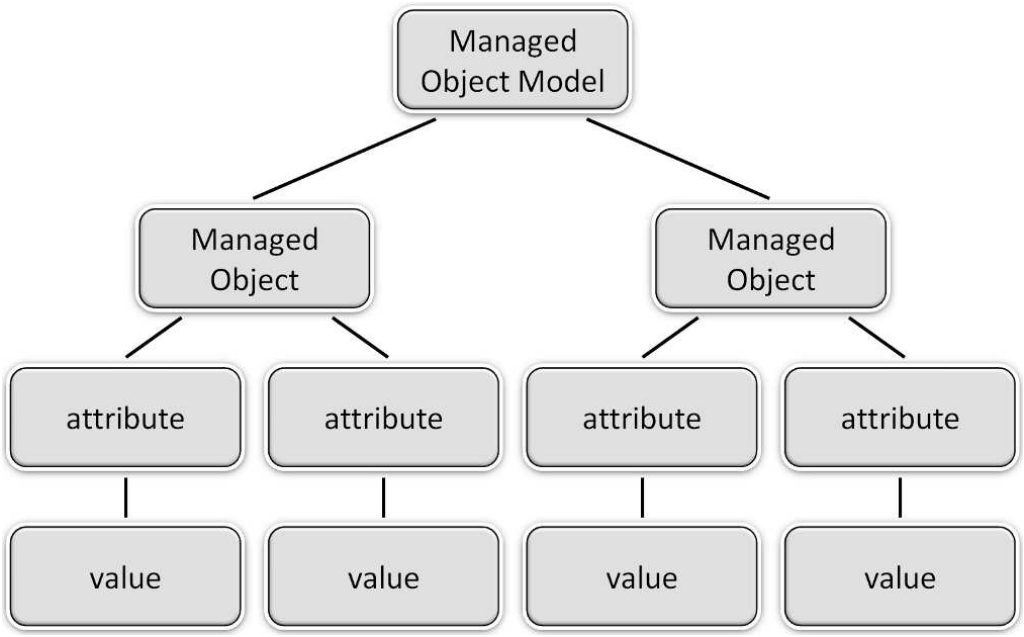

Figure 4: The simplified structure of Managed Object Model.

2.4 Mobile Switching Center Server (MSC-S)

The Mobile Switching Center Server is another key network element in the Mobile Softswitch Solution. The MSC-S is located in the control layer of the layered architecture. In general, the MSC Servers are responsible for the signalling traffic and call control, i.e. they control the traffic handling in the network [9]. This is achieved by using the Gateway Control Protocol (GCP) for the data exchange between the MSC Server and the Mobile Media Gateway. In addition, the MSC Server is responsible for e.g. bearer services, teleservices, charging and security as well as some additional supplementary services [9].

The MSC-S provides the possibility to control different types of networks, including TDM, ATM and IP-based transport networks [9]. As the MSC-S is capable of TDM, ATM and IP transport, it supports the evolution of the circuit-switched core network towards pure IP transit architecture. In addition, both GSM and WCDMA traffic can be controlled simultaneously in the same node.

The MSC-S supports various standards and a wide range of standardised protocols. For example, 3GPP standards are supported. It is also possible to introduce the MSC-S to mixed-vendor networks. [9]

The MSC Servers can be integrated into a pool in which several MSC Servers operate together in order to handle the traffic load in the network. The pool functionality may be used to balance the peak load in the network during the peak traffic hours. In addition, the MSC pool can be deployed at a central site, allowing centralised and simplified network management. This may decrease costs in operation and maintenance (O&M). The MSC pool functionality is visualised in Figure 5. [9]

In case that e.g. one of the MSC Servers in a pool needs maintenance, it can be safely disconnected from the live network during low traffic hours, because the

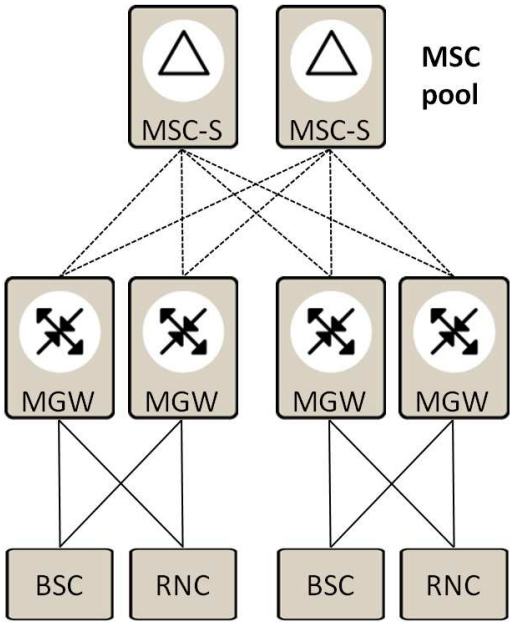

Figure 5: The MSC pool functionality.

traffic load can be distributed to the other MSC Servers within the same pool [9]. Thus, the MSC pool functionality also increases the network redundancy.

The MSC Server is scalable [9], meaning that its capacity can be increased simply by adding more hardware to the cabinet. In this way it is possible to improve the performance and efficiency. The capability for such adjustments is vital in order to meet teleoperator needs and dimensioning requirements [9]. Moreover, the next-generation MSC Server, the MSC-S Blade Cluster, offers even more capacity and improved performance. The MSC-S Blade Cluster is discussed in the next subsection.

2.4.1 MSC-S Blade Cluster

The M-MGw GMPv4 and the second generation MSC Server, the MSC-S Blade Cluster, are the latest improvements to the Ericsson Mobile Softswitch Solution. It is essential that the tool, which is developed in this thesis, is capable of handling also the latest node generations. Hence, understanding the MSC-S Blade Cluster concept is useful.

The MSC-S Blade Cluster consists of multiple advanced generic processor boards, or blades, which work together in a group or cluster. The blades run the MSC Server application, which is responsible for controlling calls and M-MGw nodes. The subscriber traffic is distributed between the available blades. [10]

The main advantages of the MSC-S Blade Cluster are improved capacity and scalability [10]. The MSC-S Blade Cluster capacity can be expanded as the traffic grows by adding new blades to the cabinet [10]. The node also has high availability, allowing O&M without causing disturbance to the network [9].

High capacity and scalability allow a simplified network topology with fewer

nodes, since it is not always needed to introduce a new MSC Server to the network, if the capacity requires an upgrade. This naturally means savings for the teleoperators. Furthermore, the MSC-S Blade Cluster can also be integrated into an MSC pool for additional network redundancy and cost-savings [9].

2.5 Other MSS-related network components

In this section, a few other MSS-related network components are discussed. These nodes are not included in the prototype, which is developed as a part of this thesis. However, support for the RNC (Radio Network Controller), BSC (Base Station Controller) and the Redback router could be integrated into the tool later on.

RNC and BSC
The Radio Network Controller (RNC) is a network element located in the Public Land Mobile Network (PLMN) [11]. Its function is to control one or more Node Bs [11]. In Ericsson's networks, the M-MGw is connected to the RNC using either ATM or IP. The RNC then controls the connected Radio Base Stations (RBSs) in the UMTS radio access network.

The Base Station Controller (BSC) is another network element in the PLMN, being responsible for controlling one or more Base Transceiver Stations (BTSs) [11]. In Ericsson's networks, the M-MGw is connected to the BSC using TDM. The RNC and BSC and their interconnections are shown in Figure 6.

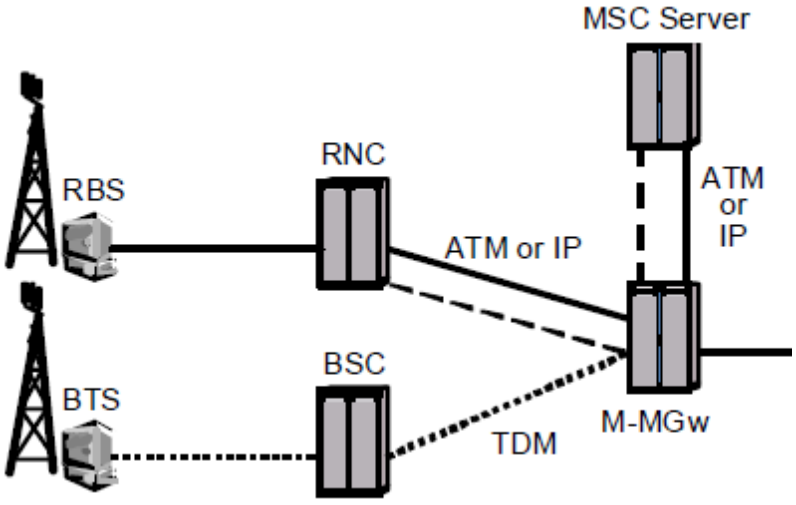

Figure 6: The M-MGw interconnections to the RNC and BSC.

HLR and VLR
The Home Location Register (HLR) is the primary database that contains the information about all subscribers in order to support session or call handling [11]. In addition, the current location of each user equipment (UE) is stored in the HLR.

The Visitor Location Register (VLR) is an entity that contains the information needed for setting up calls. Moreover, the VLR contains the information of all UEs, which are registered to the location area served by the VLR. [11]

AuC and EIR

The Authentication Center (AuC) is needed in order to authenticate mobile subscribers to the network. The AuC is associated with the HLR [11].

The Equipment Identity Register (EIR) is the entity that stores International Mobile Equipment Identities (IMEIs) [11]. The EIR may, for example, contain a list of IMEIs, which have been banned from the network.

Redback router

The Redback routers are capable of handling data, voice and video services. These routers can be found in the operator networks provided by Ericsson. If the prototype development is continued after this thesis, it is likely that these routers are included in the tool. This is because the routers are needed in order to provide a full overview of the network configuration.

3 Network configuration and supervision tools

This chapter gives an overview of the most important network configuration and supervision tools that are used today. Not all tools are covered, however, as the emphasis is on tools related to the M-MGw and MSC-S. First, certain node management tools are introduced, after which the focus is more on network level tools.

As will be shown in this chapter, no common tool exists capable of handling the major tasks regarding configuration and auditing on a network level. Thus, the tool that is developed as a part of this thesis has the potential to give extra value in terms of network level auditing and configuration.

3.1 Node Manager

Node Manager is a tool that provides a graphical user interface (GUI) for the O&M (Operation and Maintenance) functions in the M-MGw nodes. Node Manager is delivered as a part of M-MGw software releases. In general, it contains all the functionality that is needed for operating an M-MGw node. The node management is performed by reading and manipulating the Managed Objects (MOs). [12]

The main tasks that Node Manager can handle are configuration management, software and hardware management as well as fault management [12]. Node Manager can therefore help troubleshooting and node performance evaluation. Configuration management makes it possible to create, delete and modify Managed Objects either manually or with automated MO scripts. Software versions can be managed with node upgrades. In addition, the executing software version can be changed easily with Node Manager. Hardware management is possible with a graphical presentation of the installed hardware. The GUI is also capable of showing the current statuses of the hardware. Finally, the alarm view and the event log are useful in terms of fault management.

Node Manager is written in Java, thus it can be run on almost any computer provided that the Java virtual machine (JVM) has been installed. Node Manager makes it possible to manage a node both locally and remotely, thus enabling efficient and centralised node management.

The main advantages of Node Manager are the ability to hide the inner complexity of the node database and the ability to show the structure of the Managed Object Model (MOM) in a user-friendly format. Hence, Node Manager is suitable also for less experienced end-users.

Perhaps the main disadvantage of Node Manager is a certain lack of efficiency. Several actions might be required in order to get work done, i.e. the tool can be slow and inefficient in some cases. However, M-MGw nodes can also be managed with a text-based node management tool called MoShell, which can be more efficient, depending on the task. The next section introduces MoShell in more detail.

3.2 MoShell

MoShell is a text-based network element manager for nodes that operate on the so-called Cello platform. These nodes include the M-MGw and RNC, for example. In general, MoShell is capable of reading and editing Managed Objects, therefore it provides similar O&M services as the M-MGw Node Manager.

The main functionalities of MoShell include MO configuration, alarm service, performance management and logging service. All of these functionalities are accessed via the command-line interface. MO configuration allows performing various MO operations, including reading MO contents, setting attribute values, creating and deleting Managed Objects as well as executing different MO actions. The alarm service provides access to the active alarm list, whereas the performance management makes it possible to scan the performance counters in the node. The logging service provides alarm, event, system and availability logs. Such logs can be useful in many different tasks, like in troubleshooting.

MoShell can be used on multiple platforms, i.e. Solaris, Linux and Windows are all supported. It is also possible to run Unix commands from within MoShell, thus this feature makes MoShell more versatile.

MoShell provides certain powerful features that help in node configurations, troubleshooting and in node management in general. For example, it is possible to perform operations on many Managed Objects at the same time, which is a tremendous advantage over Node Manager. MoShell also supports scripting, which gives new opportunities for node management, because many processes can be automated. It is also possible to read the whole node configuration data and save the output to a log file.

As a downside, MoShell is not really suitable for inexperienced users. The syntax is challenging at first, and learning it takes time. In other words, MoShell requires lots of competence and experience from end-users before one can take full advantage of all features. Fortunately, there is a help functionality that can solve the most common problems. Next, the following section introduces WinFIOL, which can be used for managing MSC Servers.

3.3 WinFIOL

WinFIOL is a program designed for O&M, installation and testing of the AXE communication platform [13]. WinFIOL can therefore be used for managing MSC Servers, including the latest MSC-S Blade Clusters. The program provides a standard command-line interface for end-users and supports several communication protocols, including TCP/IP.

With WinFIOL's features and functionalities it is possible to manage nodes efficiently. First of all, WinFIOL uses MML language for sending commands and receiving MML printouts. For instance, the MSC-S IP configuration can be accessed with specific MML commands. Second, the command printouts can be stored in a log file for further analysis. Third, command files are supported, which make it possible to automate data collection and to send multiple commands simultaneously.

Fourth, WinFIOL features a scripting language for enhanced and automated MML handling. Finally, WinFIOL's functionalities can be extended with optional plug-in modules. [13]

In spite of all the useful features, there are some drawbacks as well. The MML language might be difficult to cope with, even for experienced users. The commands are not always logical and can be difficult to remember. Hence, end-users might need to refer to the user guide relatively often. Finally, WinFIOL is not able to handle M-MGw nodes. This complicates the MSS (Mobile Softswitch Solution) level network management, because multiple tools are needed.

3.4 OSS-RC

The operations support system for radio and core network (OSS-RC) is a common network management system for GSM, WCDMA and next-generation LTE networks. The OSS-RC can be used for managing radio access networks (RANs) and core networks (CNs) in both circuit-switched and packet-switched domains, thus ensuring the management of complete networks. With built-in network management functionalities, the OSS-RC makes it possible to manage and optimise both nodes and network resources. [14]

The OSS-RC is a high-level program that consists of several separate tools with a common look and feel. These separate tools operate on the node level, and together they cover the management of the complete network. The OSS-RC provides a platform for launching the node-specific tools, an example of which is Node Manager. In the radio access networks, the OSS-RC enables for example the auto-provisioning of radio base stations and methods for improving the coverage [14].

The OSS-RC also provides statistics from the network, including information about alarms, faults and performance. In addition, the OSS-RC is an optional tool for M-MGw operation and maintenance (O&M). It is fully capable of performing node upgrades, like Node Manager. The OSS-RC extends, however, the Node Manager functionalities with additional performance monitoring.

As a network level tool, the OSS-RC has the potential to simplify network and node management. The statistics it can provide from the network may be useful, thus allowing the utilisation of network resources more efficiently. In addition, the integrated network management may lead to cost-savings [14].

As a downside, the OSS-RC is heavily dependent on the node level tools. It can be somewhat confusing to launch a tool from the OSS-RC that could also be used as a standalone version. Furthermore, the additional value provided by the OSS-RC may not be that significant, even though it certainly helps managing all the required network level tools. Finally, although the OSS-RC operates on the network level, it is not able to perform full-scale network auditing nor supervision. In the following section, the Nemo tool is discussed, which is a tool for M-MGw configuration.

3.5 Nemo tool

Nemo (Network MO) tool is a program that is used for generating the M-MGw traffic configuration. The Nemo tool creates MO scripts, which contain the entire network data needed for traffic handling. The MO scripts are executable by Node Manager and MoShell. [15]

In practice, the Nemo tool is an XML file to MO script converter. The XML file, which can be handled by standard spreadsheet applications, has a predefined column/row structure. The end-user makes the desired changes to the file, i.e. defines the relevant Managed Objects and sets the corresponding attribute values. The file is then parsed by the Nemo tool, which keeps track of the Managed Objects and the MO types. As an output, the traffic configuration is generated for an M-MGw node.

Perhaps the main advantage of the Nemo tool is that it can significantly simplify network configuration. The user interface is relatively easy to use, yet it is capable of creating all the necessary network data. Hence, the Nemo tool reduces the end-users' competence requirements.

As a disadvantage, the end-user must know exactly what to do, because the input given by the end-user is not validated by the tool. Thus, an incorrect input may cause the scripts to fail when they are executed. In addition, the Nemo tool only creates entire network configurations. Hence, it is mandatory to regenerate and re-execute the complete network data, even if just a small reconfiguration is needed. This may take lots of time. Finally, the Nemo tool has a fixed XML structure, thus the tool must be updated manually when new features are introduced to the M-MGw.

3.6 CCR-tool

The CCR-tool (Customer Configuration Requirement tool) supports production of node-specific configuration files in order to speed up network configuration and integration of nodes. The configuration files, produced by the CCR-tool, are syntax-checked and can be used to configure all high-volume network elements, including the M-MGw and MSC-S. [16]

Ericsson's customers and internal users work with Excel-based CCR-forms, which are used to gather the node-specific configuration data. The forms are then imported into the CCR-tool system, which is a web-based application. The system creates a project, from which configuration files can be generated. A CCR-tool project can also be created by importing configuration data from live nodes. [16]

Before performing any node configurations, it is essential to ensure that the configuration files are loadable into a physical node. This verification is done by the CCR-tool. The CCR-tool also provides network pictures, which are visual representations of the CCR-tool projects. These network pictures can be particularly useful, when node configurations are imported from a live network. [16]

The CCR-tool provides certain advantages for end-users. First, it enables fast production of node-specific configuration files. Second, it reduces competence re-

quirements in terms of configuration knowledge. Third, it improves the quality of configuration activities by providing correct syntax in the configuration files. Finally, the CCR-tool is web-based, i.e. there is no need to make any local installations prior to using the tool.

Although the CCR-tool visualises the network configuration, it is not capable of providing live status information from the nodes. Thus, the CCR-tool is not a network auditing tool.

4 Theory and methodology

This chapter presents the general theory and the methodology that are needed in the prototype development. First, an overview of prototyping is given. Understanding what prototypes are for and how to benefit from them is vital information from the thesis point of view. Second, the theory behind software requirements specification and its importance is described. Then, the software development model that is used in the prototype development is explained. At the end of this chapter, a review of two popular programming languages is presented. This information is needed when the programming language for the prototype is chosen.

4.1 Prototyping

In software development, the term prototype refers to a working model of some product or application. Typically, prototypes are used in the early phases of the development lifecycle. The idea is that the prototype should implement a subset of the given software requirements, i.e. prototype as such is not a full-scale application [17], [18]. Hence, prototypes are suitable for experimenting something new in order to gain practical experience [17]. In general, the possibility to try out different design alternatives, test theories and confirm performance issues prior to productisation is a vital component for successful software development.

An additional benefit of prototypes is their low production cost. That is why prototyping can be used frequently when needed. For example, a prototype could be created in order to show it to the customer and to get quick feedback. If it turns out that there is not enough reason to fully implement the application, the development can be easily stopped, since the used effort and resources spent are relatively small. Moreover, if some problems are found from the prototype, it is easier and cheaper to fix them early in the development phase than after productisation.

Prototypes offer a valuable learning experience, because the development of a prototype often leads to getting more insight from the actual requirements and alternative designs that were not considered beforehand [18]. In addition, the development team alone is unlikely to come up with the best possible design solution. Therefore, new ideas can be formed when the prototype is introduced to a wider public, and any feedback from it may be useful later on. These new thoughts can be used to refine the initial requirements and to shape the expectations of the whole application. As a consequence, the feedback from a prototype can transform the final product into something totally different than was initially intended.

In fact, there is a slight risk in a software development project, if no prototypes are developed [19]. Without prototyping, the application under development might not meet the users' needs [17], and an incorrect set of requirements could be satisfied [18]. This uncertainty as to whether the new design can do what is desired, is present in projects that do not rely on prototyping. In addition, it is often those unforeseen problems that may, in the worst case, lead to cancellation of the whole project. Prototyping can, however, relieve the uncertainty. Thus, prototyping can be seen as a development method that reduces risk and increases software quality [19].

One fundamental aspect that prototypes can provide, is a concrete basis for further discussions between software developers, product management and also end-users [17]. Thus, a prototype can help decision-making regarding the possible future development [17]. Moreover, prototypes can be used to prove that the product has the potential to succeed in a full-scale implementation. Getting this kind of in-depth information is, for sure, very valuable.

4.2 Software requirements specification (SRS)

It could be said that a software requirements specification (SRS) represents an organisation's opinion or understanding of a software system. An SRS is a written document that describes the system requirements, functions, capabilities and dependencies, which are usually needed before any system-level design or development can be started. A software requirements specification is an important part of software development, because it can be used as a tool for conveying information between e.g. the organisation and a client to ensure that both sides have a common understanding of the system. In case that there's no customer involved, the SRS can also be used within the same organisation, for example, between developers and product management.

The IEEE 830 standard defines the contents of a software requirements specification (SRS). In general, the standard provides recommended practices and approaches that help creating the software requirements for any type of software. It is useful to note, however, that an SRS offers no ready solutions or design suggestions. Instead, the SRS contains merely the understanding of the requirements of a specific system. [20]

A desirable SRS is written using precise expressions, so that there is no possibility for misunderstandings. Vague and too general description of the system might fail at providing the common understanding between the intended parties. Instead, a well-designed SRS describes accurately what a customer wishes to obtain, both to the software suppliers and also to the customers themselves for future reference. In addition, the SRS can assist potential users to judge whether the software meets their needs or how it should be modified in order to meet them. [20]

Moreover, an SRS decomposes the system under development into component parts, allowing easier problem solving by organising information and solidifying ideas. This, in turn, can help reducing the development effort, since the SRS forces the developers to think of the requirements before the actual design and coding phases, thus reducing the risk of redesign and recoding. Careful review of the requirements can reveal unforeseen problems from the intended design, and also provide an estimate of the required amount of work. It is useful to reveal any inconsistencies early in the development lifecycle when such problems are still relatively easy to fix. [20]

Finally, the IEEE 830 standard describes the common characteristics of a well-written SRS. These characteristics are presented in Table 2.

Table 2: The characteristics of a well-written SRS. [20]

Correct A good SRS must be correct, because any incorrect information might blur the understanding of the application. Being correct also means that the specification must be kept up-to-date.
Unambiguous Being unambiguous means that there is only one interpretation. This is important in order to ensure a common understanding between e.g. the developer and product management.
Complete The software requirements specification should include all necessary information that is required by the developers in order to create the specific application.
Consistent The SRS document should be consistent with its terms at all times. Otherwise there is a risk of causing confusion. For example, if a certain term is used to refer to a specific part of the system, the same term should be used every time when referring to the same system component.
Ranked for importance The requirements should be sorted so that the most important requirements are introduced first. In case that some high-level requirement is not necessarily achievable, it should not be on the list at all.
Verifiable The requirements should be formulated so that they can be verified. Thus, quantitative requirements are recommended, because such requirements can be measured relatively easily.
Modifiable It should not be necessary to repeat the same information in more than one place. It only makes the document harder to maintain.
Traceable Connecting the SRS to a higher level document is not always, but quite often needed. The higher level document might be e.g. a business plan that explains the purpose and the vision of the system that is under development.

In this thesis, the IEEE 830 standard is used to ensure that the most important aspects of the SRS are included in the prototype requirements. However, the standard is intended for large-scale software development projects, thus not all aspects

are necessary for the prototype development. As explained in the next section, collection of the requirements is an important phase in software development, thus the general guidelines provided by the IEEE 830 standard are needed during the prototype implementation.

4.3 The Waterfall model

In this thesis, the development of the prototype for network level configuration and auditing follows, to some extent, the so-called Waterfall model. This model is a sequential software design process that got its descriptive name from the progress that flows from the top all the way to the bottom, just like a waterfall. The Waterfall concept is illustrated in Figure 7.

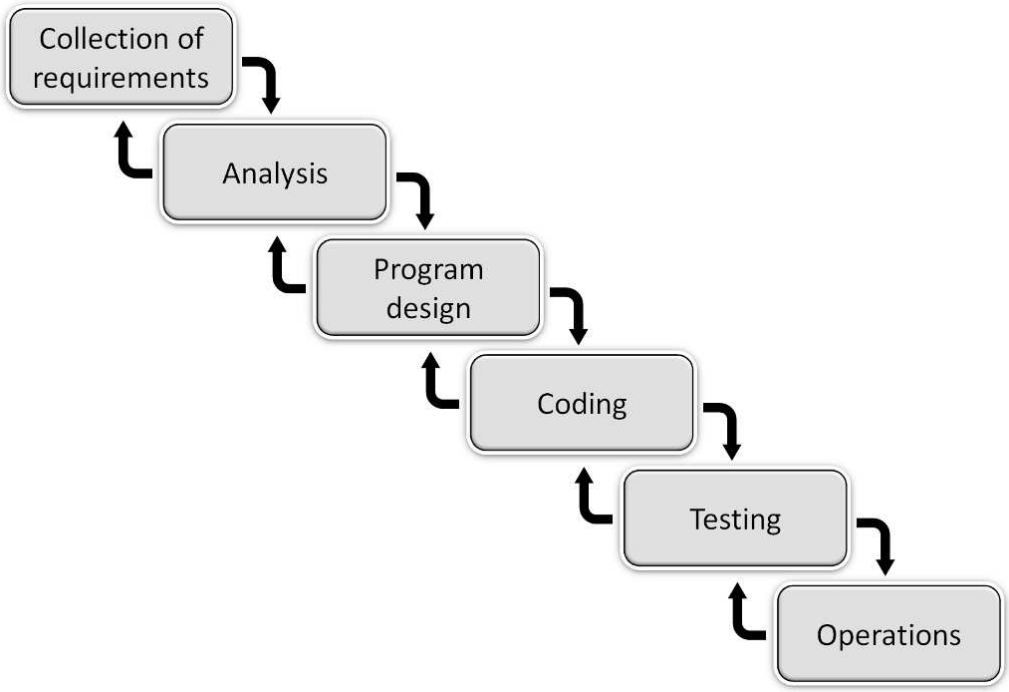

Figure 7: The Waterfall model (modified from W. W. Royce's original model [21]).

As indicated by the arrows in Figure 7, the Waterfall approach focuses on a systematic progression between different phases. Each of the phases consists of a set of activities that must be accomplished before the next phase. After completing one phase, the next one may begin, and so on. The output of the previous phase serves as an input for the following phase. [22]

The first step in the Waterfall model is to gather the requirements. In this phase it is critical to collect all requirements, because system design is based on the requirements specification. Thus, the system might not be able to satisfy the requirements that are added at later stages, which might lead to an unusable system.

After this phase is completed, the requirements specification is passed onwards to the next phase.

The second and third phases consist of analysis and design phases, respectively. The analysis phase consists of investigating the requirements that were collected in the previous phase. The results of the analysis are then passed to the actual design phase. In the design phase, the system is designed so that all requirements are taken into account. Naturally, the design can only be based on the current understanding of the requirements. If any new requirements are introduced, a step backwards in the model might be necessary. Before the actual coding can be started, it is crucial to understand what exactly will be created and what will it look like. Hence, in the design phase all the initial ideas and plans are solidified.

The fourth step in the Waterfall model consists of the actual coding. If the model has been followed successfully, the implementation phase should be relatively straightforward. This is due to the fact that all requirements and selected design principles should be known at this stage. Problems may occur, if more requirements are introduced. This, however, is against the principles of the Waterfall model. Finally, the coding phase passes the code onwards to the next phase.

The fifth and sixth phases consist of testing and operations, respectively. When the testing phase is reached, the major parts of the code have been finished. However, there might still be some minor enhancements or bug fixes that must be coded. The main goal for the testing phase is to ensure high software quality and to discover critical faults from the system. After extensive testing, the product is ready for delivery, or whatever operations it has been intended for.

In large-scale software development projects that involve several people, the Waterfall model might not be an optimal choice, because the method itself is more process-oriented rather than people-oriented. Perhaps the main drawback of the Waterfall model is its certain inflexibility. Accurate collection of the requirements at the beginning of the project is a real challenge, since they tend to change, especially during large-scale projects. In practice, customers are unable to tell everything they need in advance, which might cause problems at later phases in the development lifecycle. [22]

In the worst case, the problems encountered lead to redesign, recoding and retesting. This could happen, if a critical fault was found from the system during the testing phase, and if there was no other way to solve it other than with a completely new design. However, these kinds of problems tend to hamper large-scale rather than small-scale projects. In general, the Waterfall model is considered to adapt reasonably well to small software projects [22].

The Waterfall model might be getting a little obsolete in terms of today's software development, but for this thesis, it is not worth using any other than the basic approach provided by the Waterfall model. The reason is that the development of the prototype is a very small-scale project, and as stated earlier, the Waterfall model should be suitable for a small project like the one in this thesis. In addition, an iterative approach that some other software development models offer, is not considered necessary.

4.4 Programming languages relevant to the prototype

Before the coding phase of the prototype can be started, a decision must be made on the programming language for the prototype. Each programming language has its own characteristics, pros and cons. Thus, it is important to select a programming language that offers the best benefits for the ongoing project.

The strategic decision on the programming language could be based on the popularity of programming languages. As of today, Java and C are the most popular [23]. Thus, adoption of either of those programming languages could be considered as a safe choice for most projects. Popular programming languages often have the advantage that they are widely supported, i.e. they have support for multiple platforms and operating systems, and there is a lot of documentation available for the developers. In addition, popular programming languages are likely to have a wide range of APIs (Application Programming Interfaces) for various purposes, including graphical user interfaces, security, networking etc.

In this thesis, only Java and C are considered as possible programming languages for the prototype. The first reason for this is their popularity, which ensures long-term support for the tool under various environments. Second, selection of either Java or C guarantees more time for the prototype development, because it is not required to learn a new programming language. Finally, both Java and C offer good enough performance and support for a wide range of APIs, which can be useful during the prototype implementation.

Next, the following subsections give a brief introduction to Java and C, respectively. The selection of a suitable API for the prototype depends on the chosen user interface, thus no APIs are introduced in the following subsections. However, Section 5.8 (p. 36) presents the possible user interfaces with a brief description of suitable APIs.

4.4.1 Java

Currently, Java is the most popular programming language [23], possibly due to its simplicity and the fact that it can be run on multiple platforms. Java is used in all kinds of applications, ranging from small web applets to large commercial tools.

Java is a programming language that can be written on one platform, after which it can be run on many platforms without recompiling. This is possible, because the Java compiler compiles the source code into bytecode, which is executed by the Java Virtual Machine (JVM). Thus, the same bytecode can be run, for example, in Unix and Windows environments. [24]

The platform independence is one of Java's greatest advantages. However, the portability has its drawbacks. First of all, Java software might not run as expected on different platforms. Thus, the software might require extensive testing on each individual platform [25]. Second, the bytecode that Java uses does not offer as good performance as binary code. Many other programming languages are compiled into binary code, which can be run directly on the CPU (Central Processing Unit). In Java, the bytecode is executed by the JVM, which, in turn, runs on the CPU. This

tends to deteriorate the performance of Java, at least to some extent. Hence, Java might not be the best choice for applications that require high performance.

By definition, Java is an object-oriented language [24]. Java objects have certain specified state information and certain behaviour, thus they are similar and analogous to real-world objects. Objects are used to describe some logical part of the software, and if two objects are of the same kind, it is said that they are instances of the same class.

Java has built-in garbage collection, i.e. the programmer does not need to worry about allocating and de-allocating memory. Instead, the Java runtime environment takes care of the memory management. This feature is convenient for programmers, but the drawback of garbage collection is greater memory consumption and slower runtime speed. [24]

Another strength of Java is its API (Application Programming Interface) that promotes reusing code. After all, there is not much point to rewrite code that someone has already written and tested. In addition, Java's built-in capabilities include lots of useful functions, such as support for graphical user interfaces (GUIs). [24]

All in all, Java is a relatively high-level language, because the programmer need not consider memory allocations. This makes Java suitable for e.g. quick prototyping. In addition, Java's simplicity and portability are highly useful features, which can partially explain Java's popularity.

4.4.2 C

In contrast to Java, C is a relatively low-level language. This means that C handles similar types of objects to those that most computers do, with no possibility to deal directly with advanced data types, such as character strings, arrays or lists. This also means that there is no garbage collection, like in Java. However, C supports a comprehensive amount of operators and data structures, and features modern flow control, thus allowing C to adapt to all kinds of needs. [26, p. 5–7]

C features a block structure that encourages well-structured programs. The software structure also includes functions that are the containers for executable code. Function calls are used to provide higher-level capabilities. For example, functions can be used to derive new data types from the built-in low-level data types. Due to the popularity and evolution of C language, a standard set of functions have been developed. This so-called standard library provides a reasonable amount of features, allowing more advanced techniques to be used. [27]

Although C is a relatively simple and minimalistic language, it is not tied to any particular computer architecture. On the contrary, C was developed for cross-platform programming in spite of its low-level capabilities. Thus, C makes it possible to create programming applications that are portable across a wide range of platforms. [26, p. 5–7]

In terms of performance, C is an effective language that offers superior execution speed [26, p. 5–7]. This is due to C's straightforward architecture that allows the programmers to operate close to the underlying computer [27]. Another reason explaining C's good performance is the use of simple compilers, which compile the

source code directly into binary code.

To conclude, it could be said that good performance combined with cross-platform capabilities are the key factors explaining the success of C. In general, C is perhaps more efficient and faster than Java. However, from the thesis point of view, there should be no significant difference in the performance, because the prototype for network level configuration and auditing is a relatively small-scale application. In other words, both Java and C are suitable for the prototype in terms of performance. Hence, one of the key factors in the programming language selection is the support for suitable APIs.

5 Tool for network configuration and auditing

This chapter focuses on introducing the prototype for network level configuration and auditing. First, the current challenges regarding network configuration and auditing are presented, followed by a solution proposal that could make it somewhat easier. The third section introduces the purpose for the prototype. In the fourth section, some possible use cases for the tool are identified, including the target audience. Then, the focus area of the prototype is described, followed by a section that explains the operational principle and challenges of the prototype. After that, the requirements for the tool are presented. The requirements specification is divided into must-have and nice-to-have features. Section 5.8 (p. 36) describes the solution alternatives that are examined before implementing the prototype. Finally, the process explaining the prototype evaluation is presented at the end of the chapter.

5.1 Current situation

Configuring and auditing networks can be challenging tasks in modern communication networks. Network configuration is necessary when e.g. new nodes are introduced to the network. Reconfigurations, on the other hand, are needed when some network parameters need to be modified, for example due to some problem in the network. In both cases, it is essential to get the network fully operational in a timely manner, without causing downtime that could decrease revenues for the operator.

Network auditing, on the other hand, is important in order to provide reliable service with as little disturbances as possible. The ultimate goal of network auditing is that all problems or issues are discovered and solved well before they escalate to a larger scale. For example, with proper network supervision, a hardware fault would be detected and fixed much earlier than without any supervision at all. In general, proactive network auditing would help in minimising network downtime and congestion, and at the same time, end-user satisfaction could also be improved.

As of today, no common tool exists which is capable of handling the major tasks regarding configuration and auditing on a network level. Hence, engineers analysing and configuring operator core networks need to rely on several different tools, because almost every different network node requires its own tool. This makes network configuration a time-consuming task, since configuration data must be collected from several locations using multiple tools. Also network auditing is rather challenging, mostly due to the same reason. Because of these challenges, getting an overview of the network structure might be difficult. This, in turn, can potentially lead to misconfigurations that may have a negative impact on the network. In addition, finding some small error in the configuration might take lots of time without a clear view of the network. The following section offers a possible solution to the challenges that are faced today.

5.2 Solution to the current challenges

As stated in the previous section, several tools are currently needed for network level auditing. Thus, network auditing is anything but easy and user-friendly. It is clear that network auditing should be made somewhat easier, since there is no possibility to gather all network data together for a clear overview of the network. The same applies to network configuration, since mastering several tools requires a lot of unnecessary competence, which should not be the case.

In this thesis, development of a network management tool is offered as a solution to the current challenges. This is because there is an identified need for a tool capable of configuring and supervising networks. A network level tool should, obviously, support several node types, minimising the number of tools that are currently needed in network level configuration and auditing. Having all the necessary network level functions in one clever tool would also mean less manual work in terms of network management.

The possibility to make configurations to the network elements combined with network level auditing capabilities form the basic requirements for the tool. These high-level requirements could be satisfied with a proper tool that is intended just for those specific purposes. The vision is that such a tool would solve the challenges that are faced today in network level configuration and auditing, at least to some extent.

In the scope of this thesis, however, is only to implement a prototype. After the thesis, the development of the tool might continue, if decided so by the product management. The idea is that the prototype should be capable of network level configuration and auditing, but only to some extent. As it is a prototype, it will have limited features. For example, only a few key nodes are to be supported, instead of all UMTS core network elements. More exact requirements for the prototype are introduced in Section 5.7 (p. 33).

It would be highly beneficial if the development process of the prototype could give information about the feasibility of the tool. This is one particularly important goal for the prototype and its development. Later on, this kind of information could be used when making decisions on a possible full-scale implementation. In the next section, the purpose for the prototype is presented in more detail.

5.3 Purpose for the prototype

Perhaps the main purpose for the network level configuration and auditing prototype is basically to show that it can be done, since new designs may have problems that were totally unexpected. Without any further knowledge on this matter, future plans cannot be even considered. A successful prototype might also convince product management of the necessity and opportunities of the tool. In general, prototyping may offer a great learning experience and a good impression of what can be achieved. This, in turn, can help getting the required funding and resources that are often needed for a full-scale implementation.

Another fundamental purpose for the prototype and its development is finding

answer to the research problem of the thesis as well as reaching the rest of the thesis goals. The research problem is to evaluate the feasibility of the prototype, i.e. find out how well it adapts to combining network level configuration and auditing functionalities. Such an evaluation must be done before any decisions on a full-scale implementation can be made. The other goals that the prototype can hopefully answer include e.g. whether the tool is capable of simplifying or speeding up the tasks that involve network configuration and auditing. Section 1.2 (p. 1) presents a more detailed description regarding the research problem and thesis goals.

As stated in Section 4.1 (p. 20), the building of a prototype is an essential part of practically any tool development. Therefore, a prototype for network level configuration and auditing is considered necessary. In this thesis, an additional purpose for the prototype is to get information about any issues that must be tackled prior to a possible full-scale implementation. Also the practical experience from the tool development may turn out to be valuable later on. It is vital to get that kind of information during the prototyping phase of the project, because in case that there are major changes that need making, they can be implemented in the later versions at a reasonable cost.

5.4 Use cases and target audience

In general, the tool for network level configuration and auditing could be used in various situations related to network management. The tool could be used, for example, to locate misconfigurations or other problems in the network, to supervise the network to make sure everything works as planned and to perform reconfigurations due to node migration or malfunctioning hardware. The aim is that the tool could provide a quick overview of the network, enabling fast and efficient task handling.

The prototype is targeted only towards Ericsson's internal use. Thus, the end-users are engineers who work with various tasks at Ericsson. The prototype could be used to help configuring and auditing test networks that often need to be updated to satisfy new needs. The prototype could also be used to help solving customer service requests, hopefully faster than before. With the prototype, a customer's network could be audited remotely without compromising network security, because only node configuration data from the network is needed, i.e. physical access to the network is not necessary. In this case, the configuration data must be fetched and provided by the operator. Moreover, performing reconfigurations is not possible without access to the network, but the configuration scripts could be easily distributed to the customer.

Good customer service in terms of quick problem resolution is something that is worth trying to achieve, and its importance must not be underestimated. Thus, another use case for the tool could be creating extra value to the customers with improved problem resolution. In general, over the past few years new business models have emerged that are focusing more and more on services, rather than on product sales [28]. The shift from products to services has been due to declining revenues from product sales and due to increasing revenues generated by services [28]. The ongoing trend has caused also the telecommunications equipment vendors

to seek for new sources of income by emphasising services more than before. Thus, there is a chance that the tool for network level configuration and auditing could contribute positively to Ericsson's service portfolio.

The prototype as such is not to be released for users outside of Ericsson. However, the productisation of the tool is a future vision. Before the vision can become reality, the tool must be implemented on full-scale. After that, the tool could be distributed also to the customers and end-users who work for various operators. Then, each operator could audit and configure their own networks using the tool. In the next section, the focus area of the network level configuration and auditing prototype is introduced.

5.5 Scope of the prototype

As of today, at Ericsson there is an identified need for a network level configuration and auditing tool. However, before a full-scale implementation can be taken into consideration, the usefulness of the tool must be evaluated. For this purpose, a prototype is an ideal choice, and this is also the reason why a prototype is developed as a part of this thesis. Furthermore, prototyping can provide a lot of additional information that can be useful later on. As explained in Section 4.1 (p. 20), one of the benefits is the opportunity of discovering new requirements. However, in order to take full advantage of the prototype, its scope must be carefully considered.

The prototype that is developed in this thesis is limited to supporting Mobile Media Gateways (M-MGws) and Mobile Switching Centers (MSCs). There are three main reasons for supporting only M-MGw and MSC nodes. First, both node types are key elements in the Ericsson Mobile Softswitch Solution. Second, it was thought that the usefulness of the network level configuration and auditing tool can be evaluated using only the M-MGw and MSC. Finally, the third reason is Ericsson Finland's involvement in M-MGw development.

The prototype should be capable of M-MGw configuration and auditing as well as MSC auditing on an IP level. As explained in Section 2.2 (p. 6), support for IP (Internet Protocol) guarantees a future-proof solution. This makes up the first reason for choosing IP-level support for the prototype. The second reason is Ericsson's need for an IP-level configuration and auditing tool. And finally, it was considered that no other bearers are needed in order to evaluate the usefulness of the tool.

In future, the tool might be extended to support all Ericsson node types as well as routers and switches. For instance, support could be added for the Radio Network Controller (RNC) and Base Station Controller (BSC). Also, other bearers, such as ATM (Asynchronous Transfer Mode) and TDM (Time Division Multiplexing) could be introduced into the tool, since they are still widely used in operator networks. From a prototype design perspective, attention must be paid to future-proof and modular design, allowing the addition of new bearers and node types.

5.6 Operational principle and technical challenges

The high-level operational principle of the prototype is shown in Figure 8. The idea is to combine the configuration data from various nodes and to provide an overview of the network and its status in a user-friendly format. In addition, the prototype should be capable of performing reconfigurations. When reconfigurations are performed, the tool provides configuration scripts. Each script is then executed in order to modify the network level parameters of the corresponding node. After that, the new configuration can be used as an input for another round of network level auditing and configuration.

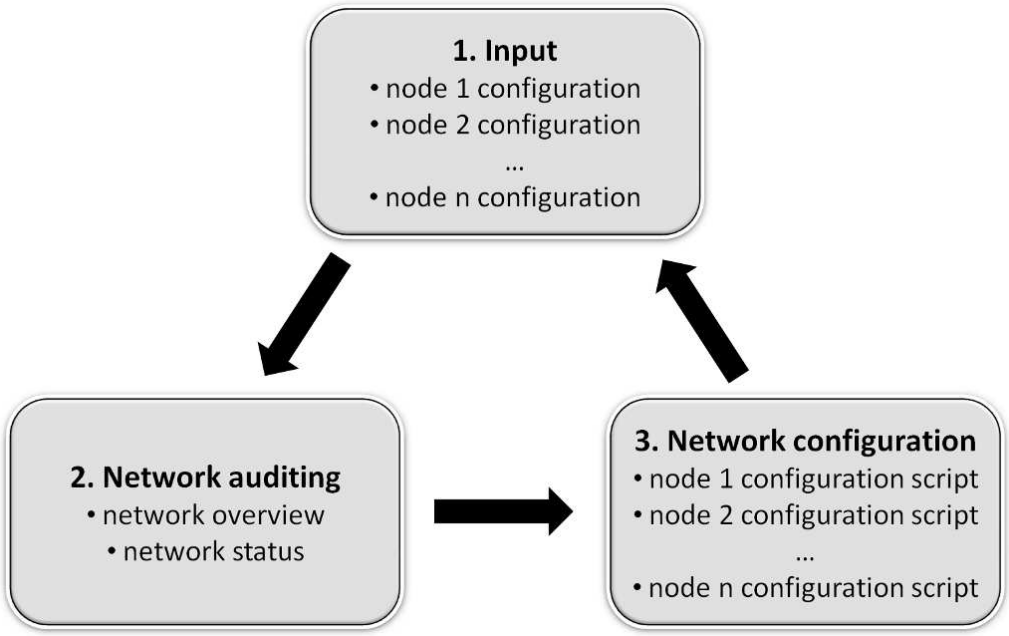

Figure 8: The high-level operational principle of the prototype.

In step 1, the node configuration data is fetched from the network elements. In the case of the M-MGw, MoShell is used to fetch all Managed Objects. When the data is fetched from the MSC, either WinFIOL or a remote connection using SSH or Telnet can be used. It is a simple procedure to get all necessary configuration data from the nodes, because a few commands merely need to be executed. In the end, all network configuration data is stored in files, each of which contains the configuration from one node. These files are then handled by the prototype. To keep it simple, automatic data fetching is not considered during the prototype development. However, automatic data fetching would make network level auditing far more convenient, thus it should be considered as a possible future improvement.

In step 2, the tool is used to parse the configuration data. In this step, the configuration data is manipulated so that the IP configuration of the network can be shown in a user-friendly format. Then, the user interface of the tool allows end-users to audit the network and its status, and to make reconfigurations.

In step 3, the tool creates the configuration scripts based on the end-users actions. The scripts are just standard text files that include commands for each network element. After the scripts have been created, they must be executed in the node in order to make any actual changes to the node configuration. The reconfigurations of the M-MGw can be done with MoShell or Node Manager. Again, the scripts must be executed manually, but automatic configuration can be seen as a possible future improvement.

The operational principle of the prototype has its technical challenges. First, the prototype is dependent on other tools. There is no guarantee that the output of e.g. MoShell will remain as it is. In its future releases, even a slight change in the output format could cause incorrect behaviour of the data parser. Thus, the need for irregular updates is a possibility. Second, the prototype is neither automatic nor real-time. This means that the end-user must fetch the data from the nodes manually, after which the network can be audited. Hence, the network overview and status are not based on the present time instant. In fact, the network overview and status are shown as they were when the node configurations were fetched. Thus, the prototype has limited capability to show the current network status that might have changed after the configuration data was last fetched. Third, after performing configurations, the data from the network must be fetched again in order to audit the new network configuration. This can take a significant amount of time, which should not be the case. The success of the reconfiguration should be confirmed by the tool immediately.

It should be noted that there are several alternatives to create a prototype for network level configuration and auditing. The prototype that is developed in this thesis, represents just one of them. Before implementing the tool fully, a feasibility study is recommended in order to analyse the strengths and weaknesses of the given operational principle, because it is possible that even better alternatives are found. Next, the following section describes the requirements for the prototype.

5.7 Requirements for the prototype

In this thesis, the prototype development follows the Waterfall software development model. As suggested in Section 4.3 (p. 23), collection of the requirements is the first and a highly important step in the development process. Thus, the IEEE 830 standard is used as a guidance for formulating the requirements for the prototype. It is expected that the IEEE 830 standard could help providing a solid basis for the rest of the tool implementation project.

However, the IEEE 830 standard is not followed entirely, since the standard is meant for much larger software projects than the project in this thesis. In addition, as the prototype for network level configuration and auditing concentrates only on its core functionalities, there is not enough reason to follow the standard entirely at all times, but rather to take advantage of its major guidelines.

As a prototype is developed in this thesis, it will have limited features. Thus, the requirements for the prototype may focus entirely on the main functionalities of the network level configuration and auditing tool. For example, the prototype

supports only Mobile Media Gateways (M-MGws) and Mobile Switching Centers (MSCs), both of which are key nodes in the mobile core network. In fact, for the prototype it is enough to be capable of M-MGw configuration and auditing as well as MSC auditing. The detailed explanation for choosing only the M-MGw and MSC for the prototype is given in Section 5.5 (p. 31).

Another important aspect of the requirements specification is its focus on the possible full-scale implementation. For example, the prototype design must take into account that new node types could be introduced to the tool. Support for all network elements is mandatory for any future releases, because all nodes are needed for the complete network level view. There is also a requirement for OS (Operating System) independence, which is essential, because the end-users would most likely use the tool on multiple platforms.

In this thesis, the identified requirements are collected into two separate lists, which together form the requirements specification for the prototype. The first list includes the mandatory requirements for the prototype, whereas the second list contains optional requirements. Optional requirements consist of features that would give extra value for the prototype. However, those features are not considered mandatory as such. It should be noted that the requirements specification is based on the requirements that are gathered before the actual design and coding phases. Hence, some changes are possible.

It is worth pointing out that the requirements that are presented in this section do not offer any ready solutions or design suggestions. Instead, the specification serves as an input for requirements analysis. After the analysis phase, the actual design of the tool may begin. In the following subsections, the must-have and nice-to-have requirements are presented.

5.7.1 Must-have features

The requirements that must be included in the prototype are presented in Table 3. These requirements can be divided into three categories. The first category describes the requirements for network overview and auditing, whereas the second category identifies the requirements for network configuration. The third category includes the requirements for the general tool design.

Requirement 1.0 states that the prototype must be able to provide an overview of the network configuration. This requirement defines the whole starting point for the tool that is developed in this thesis. In requirement 1.1 the usability aspect of the tool is considered, as the prototype must be able to convert the node configuration data to a readable and user-friendly interpretation of the network structure. Providing a logical view of the network is the key factor that determines the success for the tool.

Requirement 1.2 states that network level auditing must be possible for both the M-MGw and MSC, and as requirement 1.3 explains, the prototype must also show the interconnections between the nodes. Finally, requirement 1.4 tells that the prototype must show the link and interface statuses, which are important in terms of problem resolution. In general, requirement 1.4 is absolutely necessary for

Table 3: The must-have requirements for the prototype.

No.	Requirement
1.0	The prototype must be able to provide an overview of the network configuration on an IP level.
1.1	The overview of the network must be in a user-friendly format.
1.2	There must be a possibility for M-MGw and MSC auditing.
1.3	The prototype must be able to show the interconnections between network elements.
1.4	There must be a possibility to view the link and interface statuses (with e.g. colour coding).
2.0	The prototype must support reconfigurations of M-MGw on an IP level.
2.1	There must be a possibility to create configuration scripts for the M-MGw (add, delete and modify scripts).
2.2	The prototype must be able to show the default and recommended values for the chosen MO attributes.
3.0	The design must be OS independent, i.e. the prototype must work on all major operating systems.
3.1	The design must be future-proof, i.e. the addition of new bearers and node types must be possible.
3.2	The design of the prototype must be done on MO level to ensure understanding of the node structure.
3.3	The prototype must support the latest M-MGw and MSC software releases.

network level supervision.

Requirement 2.0 states that the prototype must support reconfigurations of the M-MGw on an IP level. This ability to make configurations, combined with auditing capabilities, makes the tool suitable for several different situations. The reconfigurations are done using configuration scripts. According to requirement 2.1, the prototype must support configuration scripts for creating, deleting and modifying Managed Objects (MOs).

Requirement 2.2 is about showing the default and recommended values for the M-MGw MO attributes. This feature is primarily needed in reconfigurations, because when network parameters are modified, it is useful to know the default as well as recommended values for each MO attribute. The values are useful also in problem resolution, because network level issues may occur if some attribute value has been set incorrectly. Thus, knowing the default or recommended value for each MO attribute might help solving the problem. Requirement 2.2 states also that the MO attribute view must be adjustable based on the user's selections, thus allowing the user to see only what is relevant. As there are different end-users each with their own context of use, it is important that they have the chance to configure the tool for their own needs.

Requirement 3.0 says that the prototype must work on all major operating systems. Furthermore, it would beneficial if the prototype worked out of the box, i.e. without need for any installations. Requirement 3.1 states that the prototype must be designed so that addition of new node types or bearers is a relatively simple task. Requirement 3.2 tells that the prototype must have the complete MOM (Managed Object Model) view of each M-MGw node, because understanding the node structure without it is practically impossible. Finally, requirement 3.3 states that the latest software releases of the M-MGw and MSC must be supported.

5.7.2 Nice-to-have features

The nice-to-have requirements are given in Table 4. Even though these requirements are not mandatory for the prototype, they are still considered highly beneficial. Besides, these requirements are mandatory in the possible full-scale implementation of the network level configuration and auditing tool.

Table 4: The nice-to-have requirements for the prototype.

No.	Requirement
4.0	The prototype should support all M-MGw and MSC software releases.
4.1	The prototype should be designed so that there would little or no maintenance work needed when new M-MGw and MSC software versions are released.
4.2	The prototype should support open file formats.

According to requirement 4.0 the prototype should work on all current and future M-MGw and MSC releases. Requirement 4.1 states that only little or no maintenance work should be needed when new software versions are released. These requirements are essential for full-scale implementation because of a large number of different software releases that are being used. There should be only one version of the network level configuration and auditing tool, and it should work in all networks, no matter what software releases are used. This would make tool maintenance a much easier task. It is also important that the maintenance work due to new node releases is relatively simple.

Requirement 4.2 states that open file formats should be supported. This is particularly important for full-scale implementation in order to avoid becoming dependent on a commercial platform or application. In future, the tool might be released to customers, and it is possible that the end-users are using other than proprietary applications and platforms. Thus, an open file format is the key to success.

5.8 Design alternatives based on the requirements

Before the implementation of the prototype can be started, it must be decided what is the way forward. As a new type of a tool is under development in this thesis,

there should be more than just one alternative for the user interface (UI) design. The importance of the UI selection must not be underestimated, because choosing an alternative that does not satisfy all requirements may cause delays and waste a lot of time.

The thing that makes the decision slightly more challenging, is the fact that the requirements might change during the implementation phase. Obviously, a lot of attention must be paid to getting all the requirements before any major decisions are made. The changing requirements may still bring some uncertainty into the project. From the prototype point of view, however, discovering such unforeseen requirements would be really useful. In the end, the selection from the alternatives can only be based on current understanding and on the set of requirements that are available at that moment.

Before starting to work on this thesis, there was quite a clear vision of what the network level configuration and auditing tool should look like. This vision was shared by Tomas Nordman and Christer Hamberg, both of whom are engineers at Ericsson Finland. The vision was to use a spreadsheet application as the primary user interface. Thus, the selection of spreadsheet as the basis for the prototype was recommended. Considering other options was, however, possible and also highly recommended. [29]

It should be noted that an exhaustive search for all possible design alternatives is neither feasible nor possible because of time constraints. The design alternatives that are presented in the following subsections, are also the ones that are considered when selecting the final design. Thus, the alternatives are described on a more general level, leaving some room for further enhancements or adjustments that may affect the final design during the implementation phase.

The following subsections introduce the UI alternatives for the prototype. A brief description of suitable APIs (Application Programming Interfaces) is also given. This is important, because an appropriate API could substantially ease the implementation of the prototype.

5.8.1 Spreadsheet as user interface

The primary alternative for the tool design is to use spreadsheet as the user interface. This alternative is recommended by engineers at Ericsson. With this design, it should be relatively easy to provide the network overview in a user-friendly format. Moreover, spreadsheets are widely used worldwide. Hence, the majority of end-users should be already quite familiar with the user interface, which is a great advantage.

The spreadsheet, with its two-dimensional grid, offers a rather intuitive user interface for displaying complicated IP-level information. The built-in grid makes it possible to take advantage of cells' relative positions and other spatial concepts, thus making it easier to understand the IP-level structure of each node. In addition, each network element could be placed on separate sheets, allowing the end-user to concentrate on one node at a time.

The spreadsheet design makes it possible to use different colours, lines and other styles to provide visual hints about the meaning of different elements, i.e. enhancing

the user experience is relatively simple. For example, the operational state of certain links or interfaces could be indicated with green and red colours, and different elements could be separated with lines, indicating that the elements do not belong together.

Another advantage of the spreadsheet is the fact that the network configuration could be saved simply to a single file. Hence, sharing the file and information about the network would be really convenient. In addition, if the file was saved in a standard file format, the spreadsheet could be opened and modified on all major operating systems as well as with several office software suites, both open-source and proprietary. In other words, there would be no limitations when it comes to working with multiple platforms.

As a downside, the user interface of the spreadsheet might be a limiting factor in some cases. Unless advanced features are used, there is a practical limit for what can be expressed using the spreadsheet UI. However, the capability of spreadsheets can be extended with macros. For example, VBA (Visual Basic for Applications) for Microsoft Office enables developers to automate processes and to create functions for various purposes [30, p. 11–15]. It must be noted that including built-in macros to a spreadsheet, which is generated automatically from node configuration data, can be a real challenge.

There are a few APIs for both Java and C that are capable of spreadsheet handling. For Java, there is at least JExcel (`http://jexcelapi.sourceforge.net`) and Apache POI (`http://poi.apache.org`), both of which enable reading, writing and modifying spreadsheets dynamically. Both APIs support the Microsoft Excel's XLS file format, a common spreadsheet file format that can be handled by multiple spreadsheet applications, both open-source and proprietary. JExcel and Apache POI use a free software license.

For the C programming language, there is at least a multi-platform library called xlsLib (`http://xlslib.sourceforge.net`), which is capable of spreadsheet handling in XLS format. The xlsLib library is free software.

5.8.2 Development of a new user interface

Development of a totally new type of UI (User Interface) is considered as a secondary choice. For this reason, no detailed design proposals are introduced in this subsection. Development of a new UI is considered only if severe drawbacks are found from the spreadsheet design. This would require that some of the must-have requirements could not be satisfied. Re-evaluation could also be necessary if some other major issues were found from the spreadsheet design. In short, the main drawback of developing a new UI is the amount of time that would be required. Due to the time constraints, this alternative is considered as a secondary choice.

As stated above, development of a new UI would require lots of time, in fact, much more than the spreadsheet design alternative. A new user interface would also require extensive pre-analysis and some usability testing. Furthermore, it is not always practical to spend too much time on designing a new UI for a prototype, especially if the goals of the prototype could also be reached with a simple yet

effective design. After all, the main purpose for the prototype is not to offer top-notch design and usability, but rather to experiment its usefulness and to show that such an application can be created. Later on, when the tool for network level configuration and auditing is implemented properly, it is a good idea to spend more time polishing the user interface for an improved user experience.

As an advantage, the new type of UI could offer practically unlimited number of design alternatives that would certainly satisfy almost any requirement. Thus, the new type of UI could overcome the limitations that the spreadsheet has. Similarly to the spreadsheet, the new type of UI could also be used on multiple platforms, if properly implemented.

If a new graphical user interface was developed, Java would be preferable to C. Java has powerful widget toolkits for GUI handling, such as Swing or the earlier AWT (Abstract Window Toolkit). Swing alone should provide all that is needed for the prototype. However, there is a feature rich toolkit also for the C programming language that could be used. It is called GTK+ (`http://www.gtk.org`). GTK+ is licensed under a free software license.

Finally, if the prototype for network level configuration and auditing was developed with a new type of user interface, it would be unlikely that all tool requirements could be satisfied in the scope of this thesis. Next, the tool evaluation process is described in the following section.

5.9 Prototype evaluation

As stated in Section 4.3 (p. 23), the Waterfall model is used during the prototype development. However, the testing phase of the prototype is not as extensive as recommended by the Waterfall model. The reason for this is that a prototype is merely under development, i.e. the prototype will not be released as a ready product as such. The prototype will instead go through tool evaluation.

First, the prototype is tested against the requirements to ensure that all the requirements have been included in the tool. Then, the prototype can be used to see how well it performs in different test networks. At first, it is possible that some problems are encountered, thus some time must be reserved for fine-tuning the prototype.

The prototype itself will be available for Ericsson's internal use without any restrictions, i.e. everyone willing to try may participate in the tool evaluation. In order to get started, some learning or demonstration sessions might be arranged in order to go through the basic functionalities and limitations of the prototype. It is also essential to show how the configuration data is fetched from the network elements. The sessions are arranged based on the general interest and act as the main media for sharing knowledge related to the prototype. It is unlikely that a user guide of any kind will be written just for the prototype.

After the tool has gone through evaluation at Ericsson's test environment, there should be a clear view of the strengths and weaknesses of the tool. Finding weaknesses from the prototype is positive, because it will lead to iterating the chosen design in possible future versions. Discovering new requirements would also be use-

ful. It would mean that those new requirements could be taken into account in future releases.

Finally, the next phase after tool evaluation is the decision on the full-scale implementation of the tool. This key decision is made by the product management, and the results of the tool evaluation play a key role in the decision process. Based on the decision, the tool development is either continued after the thesis, or discarded.

6 Results

This chapter presents the results regarding the tool implementation. First, the tool development lifecycle is discussed. Second, the decisions on the user interface design, programming language and API are explained. Section 6.4 (p. 44) presents the final design of the prototype, including brief descriptions of the main features. Also the strengths and weaknesses of the prototype are identified in the same section. Then, the results of the tool evaluation are discussed. Finally, the analysis of the results is presented at the end of the chapter.

6.1 Tool development lifecycle

The prototype for network level configuration and auditing was developed with the help of the Waterfall model, as described in Section 4.3 (p. 23). The model offered an organised way to steer the development process all the way from the requirements to a complete prototype, although the model was not strictly followed at all times. For example, the testing phase was not done as extensively as the model suggests, but this was intentional. Since the prototype is not released as a ready product, there was not enough reason to test it systematically on all node releases or to write comprehensive test descriptions.

In general, the Waterfall model turned out to be useful during the tool implementation, because it indicated clearly what the way forward should be. In addition, the Waterfall model enabled a practical way to keep track of the development progress, which, in turn, allowed more exact estimation of the remaining workload. This was considered highly beneficial during the tool development.

Furthermore, the Waterfall model made it easy to recognise the most important focus areas of each development phase. This helped minimising the risk of time-consuming mistakes. For example, it was essential to have all requirements prior to starting the actual design and coding phases. The tool requirements were discussed in Section 5.7 (p. 33).

If only a part of the requirements had been available and some code was written based on them, there would have been a risk that the system would be unable to satisfy the requirements that are introduced later on. This would have led to redesigning the whole system, thus making the current design and the code obsolete, at least to some extent. With the Waterfall model, however, these kinds of issues were successfully avoided during the prototype development, since there was no need to do any major changes to the carefully chosen design.

6.1.1 Development phases

The first step in the tool development was to gather the requirements before moving on to the analysis phase. The requirements were discussed in a meeting, and a written document was created afterwards. The requirement list itself did not change during the tool development, which was a significant advantage. As stated earlier, introduction of new requirements could have led to rethinking the tool design, thus lengthening the development time.

After the requirements were collected, it was time to analyse them and to make design proposals. As a result, two design alternatives were considered before the actual coding was started. The first and primary alternative was to use a spreadsheet as the user interface, while the second option was to create a totally new type of design. In the end, the spreadsheet turned out to be the most rational option. Hence, it was decided that the prototype must be based on the spreadsheet design. The next section presents more reasons explaining why the spreadsheet design was chosen.

After the design had been chosen, it was time to choose the programming language. Java and C were considered, and Java was selected for this project. Java offered several advantages, including portability from one environment to another. In addition, Java is supported in all major operating systems, thus it satisfies the requirement for OS independence. In order to create and read spreadsheets with Java, one option was to use some ready-made API (Application Programming Interface). Although other options were also considered, a Java API called Apache POI (`http://poi.apache.org`) was chosen to provide support for spreadsheet handling. The chosen programming language and API are described in more detail in Section 6.3 (p. 43).

During the coding phase, weekly meetings were organised in order to discuss how the programming phase is proceeding. The idea was to show what had happened since the previous week's meeting and also to highlight any problems that need to be solved. In the end, the meetings were arranged roughly every two weeks. Nevertheless, the meetings provided a lot of useful input, suggestions and fresh ideas about the implementation in general.

The tool was tested continuously on many different nodes during the coding phase. At least to some extent, this procedure ensured a good support for various network elements, including different configurations and software releases. However, this type of testing was not carried out on all possible configurations. In addition, the testing was done only in the local test environment, because configuration data from live networks was not available.

After the coding phase was finished, the tool was evaluated as explained in Section 5.9 (p. 39). In the tool evaluation, the tool was used at Ericsson's test networks in order to see how it performs. The results indicate that the prototype is able to audit the networks successfully, and also network level configuration seems to work relatively well. Section 6.5 (p. 48) presents the outcome of the tool evaluation in more detail.

6.2 Chosen user interface design

Based on the design alternatives that were proposed in Section 5.8 (p. 36), it was decided to use spreadsheet as the UI (User Interface) of the prototype. Prior to starting the prototype development, the spreadsheet design was considered as the best choice, and the analysis of the alternatives confirmed this.

Designing and developing a totally new type of GUI (Graphical User Interface) would have been somewhat risky, considering the limited amount of time for the

thesis. Creation of the new GUI would also have required an extensive pre-analysis and possibly some usability testing. Thus, the new type of GUI was not the best alternative for quick prototyping.

The most rational choice was therefore to select the spreadsheet as the UI of the prototype. Unlike the other choice that required a new design, the spreadsheet did not have such time constraints that would have limited its suitability for quick prototyping. The most important reason for this selection was, however, that it was able to satisfy all the requirements that were set in Section 5.7 (p. 33).

There are four additional reasons that are in favour of the spreadsheet alternative. First, the UI of spreadsheets is familiar to the majority of targeted end-users. This is due to the popularity of e.g. OpenOffice.org Calc, Microsoft Excel and Google Docs. Hence, it should not take too long to learn the basics of the tool. Furthermore, the spreadsheet design does not restrict the end-users to any specific platform. They may stick to the office software suite and operating system of their liking, no matter whether the platforms are open-source or proprietary.

Second, the spreadsheet is a good alternative from a usability point of view. The UI fits well to providing an overview of the network, i.e. using spreadsheets for network management is rather intuitive. Each sheet may be used to represent a different logical node and hyperlinks can be used to link different sheets in a simple but effective way. For instance, M-MGw IP configuration includes a number of remote IP addresses that are used for both signalling and payload. Linking of remote IP addresses to the corresponding local IP addresses enables easy navigation between nodes. The links also make it easier to understand the network configuration. More features of the prototype are presented in Section 6.4 (p. 44).

Third, there are quite a few APIs that have been developed for spreadsheet applications, some of which, are well maintained and under constant development. An adequate level of API support is one advantage of the spreadsheet design.

Finally, spreadsheet functionalities can be extended with macros. Macros consist of commands that could be used to enrich the user interface by making it more interactive. In general, macros offer a relatively easy way to enhance the user experience.

6.3 Chosen programming language and API

Java and C were the candidates in the programming language selection. Both of them meet the requirement for the OS independence, and both programming languages can be used for spreadsheet handling. In the end, Java was chosen as the programming language for the prototype because of its better portability between different platforms.

Java is widely adopted in the majority of workstations. Thus, Java often works out of the box, or at least with the minimum number of required installations. Moreover, as the targeted end-users of the tool are not tied to any specific platform, Java's platform independence and portability are useful features. For these reasons, it was logical to select Java for the prototype development. The following subsections discuss the chosen Java API and certain software requirements for the prototype.

6.3.1 Spreadsheet handling with Apache POI

The prototype for network level configuration and auditing uses a Java API called Apache POI (`http://poi.apache.org`) version 3.7 for spreadsheet handling. Apache POI supports several file formats used in office software suites, including the XLS file format which is used in the prototype.

Apache POI was chosen, because it is free software, well maintained and seems to be getting updates on a regular basis. In addition, Apache POI provides a good developer's guide, thus making it easy to get started. Currently, Apache POI does not support macros, but the support for them could, hopefully, be added in future releases.

Alternatively, the prototype could have been based on XML handling, similarly as the Nemo tool. However, the XML interface was considered more complicated than Apache POI, and there was no reason not to take advantage of a ready-made toolkit that is freely available.

6.3.2 Java and spreadsheet requirements for end-users

The prototype for network level configuration and auditing can be run on any platform provided that Java Virtual Machine (JVM) has been installed. JVM is needed for generating spreadsheets and configuration scripts. The actual source code of the prototype has been compiled with Java version 6. Hence, Java Runtime Environment (JRE) version 6 or later is required.

The prototype uses Apache POI for handling the spreadsheets in XLS file format. Hence, a spreadsheet application that can handle (read and write) the XLS file format is required in order to manage the files generated by the tool. For example, Microsoft Excel and OpenOffice.org Calc are supported by the prototype.

6.4 Proposed solution

The prototype for network level configuration and auditing follows the high-level operational principle shown in Figure 8 (p. 32). The prototype supports the configuration and auditing of Mobile Media Gateways and the auditing of MSC Servers, including the MSC-S Blade Clusters. The configuration and auditing is possible on an IP level.

The user interface of the prototype aims at being easy to understand, enabling easy network auditing and configuration. This requires certain built-in logic in order to organise the configuration data to a spreadsheet. In the case of the M-MGw, the prototype utilises the familiar structure of the Managed Object Model (MOM). The Managed Objects (MOs) are grouped based on their MO types, so that similar MOs with similar attributes and functions are always shown together. In addition, the prototype takes advantage of the hierarchical structure of the MOM by using parent-child and reserved by -relations of Managed Objects.

The user interface of the prototype resembles the UI of the Nemo tool, although Nemo is used for quite different purposes. Hence, the users of the Nemo tool can find certain similarities in the UI design, which probably makes it easy to start using

the prototype. Those end-users, who are not familiar with Nemo, might need a little more time to learn the basics.

It is worth noticing that the proposed solution is just one alternative for a network level configuration and auditing tool. By all means, it does not mean that the proposed solution is the optimal one at all. In fact, it is most likely even better alternatives exist only waiting to be discovered.

6.4.1 Functional design

The functional design of the prototype is illustrated in Appendix A (p. 55). At first, node configuration files are fetched from the network with e.g. MoShell and WinFIOL. The network data must be fetched and saved to a file manually, although scripts can be used to speed up the process. Finally, the node configuration files are added to a specific input folder, from which the prototype can find the network data. After this, the prototype is ready for network auditing.

In network auditing mode, the prototype parses each file in the input folder and creates a spreadsheet in a specific output folder. The spreadsheet, which holds the network's IP configuration, contains two or three sheets per node, as explained in the next subsection. The spreadsheet also provides the status information of different links and interfaces. The IP configuration from all network elements, together with status information, enables network level auditing.

After the IP configuration file has been created, the end-user may perform reconfigurations. In M-MGw configuration mode, the XLS-based spreadsheet is parsed in order to detect the changes that the end-user has requested. As an output, node-specific MO scripts are created to the output folder. Each script must then be executed in order to modify the network level parameters. This can be done with MoShell or Node Manager. In the next subsection, the main features of the prototype are presented.

6.4.2 Feature descriptions

In this subsection, the most important features of the prototype are presented. The following subsection explains the strengths and weaknesses of the prototype.

Nodes on separate sheets

In the prototype, each network element reserves two or three sheets, depending on the node type. M-MGw nodes have two sheets, one for the IP signalling table and the other one for default MO values. Also MSC Servers have two sheets, one for the IP data table, and the other one for the IP routing table. MSC-S Blade Clusters have one additional sheet, which is for the VLAN (Virtual Local Area Network) table.

Each node is placed on separate sheets, since this practise allows the end-users to concentrate on one node at a time. This makes it easier to get an overview of the node's configuration.

Node configuration view and status information

The IP configuration of each node is shown on the IP signalling or IP data sheet, depending on the node type. The sheets present the nodes' IP configuration as well as the status information of various links and interfaces. The prototype presents the status information with colour coding: green means enabled status, red means disabled status and orange means unstable status.

The configuration from all network elements, together with the status information, enables network level auditing. Figure 9 shows an example from the prototype, presenting a few Managed Objects and their statuses. The figure also shows the grouping of Managed Objects based on their MO type as well as some MO attributes.

Sctp			IpAccessHostGpb			
Identity	**Act**	**Rpu**	**Identity**	**Act**	**IpAddr1 IpAddr2**	**GPB**
IPAC_GPB6		5006	IPAC_GPB6		10.2.1.10	1060
					10.2.0.11	
IPAC_GPB8		5008	IPAC_GPB8	D	10.2.0.10	1080
					10.2.1.11	

Figure 9: MO statuses are indicated with colour coding.

Hyperlinks for easier navigation

The IP configuration of each network element includes remote IP addresses, which are used for data exchange between nodes. A remote IP address in one node corresponds to a local IP address in some other node. The prototype shows the interconnections between different nodes by providing a hyperlink from remote IP addresses to the corresponding local IP addresses. Hyperlinks are also used to connect Managed Objects between the IP signalling table and default values table.

In general, hyperlinks make it easier to understand the network configuration, because they allow easy navigation between nodes and sheets. Figure 10 shows an example of linked remote IP addresses on an MSC Server. The figure also shows the status information of certain MSC-S interfaces.

Node reconfigurations

The prototype makes it possible to create configuration scripts for M-MGw nodes. It is possible to generate add, delete and modify scripts by inputting text to the status fields shown in Figure 9. The letter 'D' creates a delete script for the corresponding MO as well as for the other impacted MOs. Any other input creates a new MO with the given identity or modifies the pre-existing MO. All changes to the MO attribute values are read directly from the spreadsheet.

With macros it would be possible to illustrate which MOs are affected, if a change was applied to a certain MO. For example, if a 'D' was typed into the status field of some MO, 'D' could also be shown in the status fields of all affected MOs. This type of feature can be seen as a possible future improvement.

SAID	RIP1	RIP2	MODE	RPN
EP02IETFMGW79C	10.2.2.10		PEER	3979
EP02IETFMGW79B	10.2.2.10		PEER	2979
EP02MGW211B	10.2.2.120		PEER	26311
EP02MGW211A	10.2.2.120		PEER	22011
EP02MGW79A	10.2.2.10		PEER	2905
EP02GEN10A	10.2.2.100		PEER	28792
EP12MGW79B	10.2.3.10		PEER	2905
EP22MGW75V2S	10.2.0.10	10.2.1.11	SERVER	50752
EP22MGW211C	10.2.2.120		PEER	27111
EP22BSC083	10.2.0.80		PEER	27083
EP22BSC092	10.2.0.90		PEER	27092
EP22MGW75A	10.2.0.10	10.2.1.11	PEER	22075
EP22MSC39A	10.2.0.65		PEER	2905

Figure 10: Hyperlinks are used to connect remote and local IP addresses.

Default and recommended MO attribute values

Default and recommended MO attribute values are needed mostly in reconfigurations and in troubleshooting. The prototype is capable of showing these values for all MO types and MO attributes. It is also possible to configure which MO types and MO attributes are to be shown in the default values table. This can be done by editing a configuration file that is automatically generated by the prototype.

As the end-users may choose what information is shown, the prototype can adapt better to different needs. In Figure 11, the default and recommended MO attribute values are shown for an MO type called IpInterface.

3	MO types	MO identity	Parameter 1	Parameter 2	Parameter 3	Parameter 4
45						
46	IpInterface		vid	vLan	rps	mtu
47	IpInterface	Default	1	FALSE	TRUE	1500
48	IpInterface	Recommended	-	-	-	1500
49	IpInterface	21911	30	TRUE	TRUE	1500
50	IpInterface	21912	42	TRUE	TRUE	1500
51	IpInterface	21913	20	TRUE	TRUE	1500
52	IpInterface	22011	20	TRUE	TRUE	1500
53	IpInterface	22012	1	TRUE	TRUE	1500
54	IpInterface	31911	40	TRUE	TRUE	1500
55	IpInterface	31912	30	TRUE	TRUE	1500
56	IpInterface	31913	31	TRUE	TRUE	1500
57	IpInterface	31914	20	TRUE	TRUE	1500
58	IpInterface	32011	41	TRUE	TRUE	1500
59	IpInterface	32012	1	TRUE	TRUE	1500

Figure 11: IpInterface MOs and some of their attributes.

6.4.3 Strengths and weaknesses

One strength of the prototype is that instead of processing only IP-related data, it processes all data that is fetched from M-MGw nodes. Currently, the irrelevant data is just left unused, but since the internal logic is done, the support for ATM and TDM could be added without too much work. In the case of the MSC Server, the support for new bearers requires changes to the data parser, but otherwise the built-in logic can be reused.

Throughout the tool development, the prototype worked relatively well on various M-MGw nodes with different configurations, hardware generations and software releases. Hence, the prototype supports all current M-MGw hardware generations, i.e. GMPv2, GMPv3 and GMPv4, but also R5 and R6 software releases. From an MSC Server point of view, the latest hardware and software releases are supported. The support also includes the MSC-S Blade Clusters. However, it is possible that the prototype works on older MSC-S software releases as well, but the tool has not been tested well enough to be sure.

Another strength of the prototype is that it can be used on multiple platforms with open-source software. This can be ensured as the prototype is written in Java, and the XLS file format can be handled by many office software suites.

The prototype has the advantage of showing the IP configuration with a glance. In addition, the prototype is capable of showing the interconnections between the nodes. Hence, getting an overview of the network configuration should be relatively easy. In the end, the prototype is primarily a network auditing tool, because it can provide the IP configuration together with relevant status information.

Although network level auditing is perhaps the greatest strength of the prototype, it also has its limitations. A weakness of the prototype is that it cannot provide real-time status information from the network.

Network level configuration can be seen as a secondary feature of the prototype, because the reconfiguration capabilities of the tool are limited. For instance, an obvious weakness of the prototype is that it is only capable of simple configurations. More complicated configurations must be done with some other tool, because the prototype fails to support the configuration of all MO types. However, the support for all MO types could be added to the tool, if it is considered necessary.

Finally, the prototype is designed to collect the entire network configuration in a single file. As a weakness, the file might pose a threat to network security, if the file ends up in wrong hands. If this issue is considered critical, e.g. password protection could be added to the spreadsheet. Next, the following section presents the results of the tool evaluation.

6.5 Outcome of the tool evaluation

The prototype was evaluated as described in Section 5.9 (p. 39). At first, the prototype was tested against the must-have requirements, which were presented in Table 3 (p. 35). As a result, it was confirmed that the prototype satisfies the mandatory requirements, although it was not possible to verify requirement 1.1

because of its qualitative nature.

As a part of the tool evaluation, the prototype was tested at Ericsson's test networks. As explained in the previous section, the prototype performed relatively well, as it was capable of performing network level auditing and simple reconfigurations. It would have been useful to test the tool also on live networks, but there were no opportunities to do this.

One demonstration session was arranged in order to introduce the prototype to Ericsson's engineers. The session focused on the basic features of the tool. In general, the discussion during the session provided interesting opinions about the tool and its opportunities. First of all, the tool was seen as an improvement, as it decreases manual work. The spreadsheet format was also seen as a good and familiar user interface. It was noted that spreadsheet is a relatively common format for displaying different network configurations. [31]

Some useful criticism was also given during the demonstration session. It was noted that the prototype focuses on node level, which makes it challenging to understand the entire network configuration. It was also thought that if some MO attribute values do not match to the recommended values, the tool should indicate the network level impact. Currently, the prototype only shows the plain values, without any indication of the practical impact on the network. [31]

In the demonstration session it was agreed that the main challenge of the tool is discovering the optimal way to show the network configuration. Thus, more attention should be paid to the user interface in future releases. Finally, it was discussed that the developed tool has a chance to succeed only if it can be demonstrated to actual end-users, i.e. engineers who manage operator networks. [31]

The strengths and weaknesses that were presented in the previous section, together with the feedback from the demonstration session, offer lots of useful information about the prototype. This information should be taken into account, when the tool for network level configuration and auditing is developed full-scale. Any plans or decisions regarding the tool development, however, have not been made.

6.6 Analysis of the results

In this thesis, one of the goals was to develop a functional prototype for network level configuration and auditing. This goal was reached.

The research problem of this thesis was to analyse whether it is feasible to combine network level configuration and auditing functionalities in the same tool. It was shown that it is possible to discover and solve network level problems, such as misconfigurations, using just one tool. Thus, the integration of configuration and auditing functionalities is relatively efficient. Moreover, the efficiency of these functionalities can be further improved in the full-scale implementation of the tool.

Another goal for this thesis was to find out whether the tool could simplify or speed up the tasks that involve network configuration and auditing. The prototype itself is suitable only for very limited use cases, as it supports only two node types on an IP level. The full-scale implementation of the tool, however, has potential for many more use cases, provided that its support is extended with new bearers and

node types. This could also mean that only one tool would be needed for managing the entire network. Hence, the tool for network level configuration and auditing has the potential to simplify and speed up network management.

Finally, the last goal for the thesis was to find out whether the tool is capable of providing an overview of the network configuration and status. It was shown that this is possible, because the prototype provides the configuration of each node and also shows the interconnections between the network elements. In addition, the prototype provides the status information of various links and interfaces.

All in all, it was worth developing a prototype, because a lot of information was gathered about the feasibility of the tool. The collected information may provide useful input for the full-scale implementation of the network level configuration and auditing tool.

7 Discussion

This chapter is divided into three sections. The first section gives a brief summary of this thesis. In the second section, conclusions are presented. At the end of the chapter, some suggestions for the future development are given.

7.1 Summary

In Chapter 2, Ericsson's Mobile Softswitch Solution (MSS) and its key network elements were discussed. Chapter 2 also presented the ongoing transition towards all-IP network architecture. It was concluded that IP-related competence and IP-based tools are getting increasingly important in modern communication networks.

Chapter 3 presented various network and node management tools that are currently used. It was realised that no common tool exists capable of handling the major tasks regarding configuration and auditing on a network level.

General theory and methodology were covered in Chapter 4. For instance, the Waterfall model was presented, which provided an organised way to steer the tool development process. In addition, the theory behind software requirements specification and the importance of prototyping were discussed.

In Chapter 5, the prototype for network level configuration and auditing was introduced. The prototype was designed based on the given requirements. For instance, the requirements stated that the prototype must support the Mobile Media Gateways (M-MGws) and MSC Servers on an IP level. Finally, the performance of the prototype was evaluated at the local test environment.

Chapter 6 presented the results regarding the tool implementation. It was realised that the network level configuration and auditing functionalities can be combined efficiently in the same tool. It was also noticed that the prototype may perform relatively simple tasks that involve network level auditing and configuration.

7.2 Conclusions

In this thesis, it was shown that it is possible to realise a new type of tool for network level configuration and auditing. The tool development was a success, since the goals of the prototype implementation were reached. The analysis and evaluation of the prototype provided valuable information, which may be utilised in the case that the tool development is continued after the thesis.

Only a limited number of persons participated in the tool demonstration session, in which feedback from the tool was collected. Also, the prototype was not analysed by targeted end-users, who could have validated the findings from the prototype evaluation. Hence, the reliability of the findings could be questioned, as there might be some inaccuracy when it comes to identifying the strengths and weaknesses of the tool.

Nevertheless, it was shown that the prototype for network level configuration and auditing is able to perform the tasks that were expected prior to the implementation phase. No unforeseen problems were discovered during the prototype

implementation. In addition, the prototype was designed so that its functionalities could be extended with reasonable effort.

All in all, the tool that was designed in this thesis is considered suitable for network level configuration and auditing. From a technical perspective, the full-scale version of the tool has therefore potential to succeed. The evaluation of the tool confirms these findings, since the tool was seen as an improvement to the current situation. In the next section, some suggestions for future development are given.

7.3 Suggestions for future development

First of all, a feasibility study of the prototype is recommended prior to the full-scale implementation of the tool. The study should examine whether the user interface needs to be renewed when new bearers and node types are introduced to the tool. At the same time, the usability of the proposed design could be studied. If the study was carried out with actual end-users, there would be a great opportunity to collect feedback and to promote the tool.

Another interesting topic that could be studied is automatic data fetching from network elements and perhaps automated node configuration. Automated data fetching at regular intervals would enable efficient and nearly real-time network auditing.

It is recommended to add support for macros, because macros provide a relatively easy way to improve the user interface. It is also worth analysing whether the information security of the tool should be improved. For example, including password protection to the generated spreadsheets would improve information security.

The prototype that was developed in this thesis was tested only on local test networks. It is recommended that the full-scale version of the tool is tested on live networks. Moreover, systematic testing on different network and node configurations is recommended.

References

[1] Bos, L. and Leroy, S. Toward an all-IP-based UMTS system architecture. *IEEE Network,* 2001, vol. 15, iss. 1, p. 36–45.

[2] Yang, J. and Kriaras, I. Migration to all-IP based UMTS networks. *3G Mobile Communication Technologies,* 2000, conference publication no. 471, p. 19–23.

[3] Ericsson. Efficient softswitching. Online document, 2009. Referred 14.3.2011. Available at: `http://www.ericsson.com/res/docs/whitepapers/efficient_softswitching.pdf`.

[4] Laurikainen, A. Quality of Service in Media Gateway. Master's thesis, Tampere University of Technology, Department of Electrical Engineering, Tampere, 2003.

[5] Perttula, K. M-MGw technical overview. Ericsson internal, 2008.

[6] Ericsson. Media Gateway for Mobile Networks (M-MGw). Online document. Referred 14.3.2011. Available at: `http://www.ericsson.com/ourportfolio/products/media-gateway-for-mobile-networks-m-mgw`.

[7] Ericsson. Seven reasons to use end-to-end thinking when building all-IP networks. Online document, 2009. Referred 14.3.2011. Available at: `http://www.ericsson.com/res/docs/whitepapers/end-to-end-IP-infrastructure.pdf`.

[8] Baldwin, J., Ewert, J. and Yamen, S. Evolution of the voice interconnect. *Ericsson Review,* 2010, no. 2, p. 10–15.

[9] Ericsson. MSS/MSC Server. Online document. Referred 14.3.2011. Available at: `http://www.ericsson.com/ourportfolio/products/mss-msc-server`.

[10] Mäkiniemi, P. and Scheurich, J. Ericsson MSC Server Blade Cluster. *Ericsson Review,* 2008, no. 3, p. 10–13.

[11] 3GPP TS 23.002 V10.1.1. Technical Specification Group Services and System Aspects; Network architecture (Release 10). 3rd Generation Partnership Project, 2011.

[12] Ericsson. M-MGw Node Manager user guide. Ericsson internal, 2008.

[13] Andersson, P. WinFIOL 7.1 user's guide. 2009.

[14] Ericsson. Mobile OSS. Online document. Referred 28.3.2011. Available at: `http://www.ericsson.com/ourportfolio/products/mobile-oss`.

[15] Nordman, T. Network MO (Nemo) user guide. 2008.

[16] Szakos, P. CCR-tool user guide. 2010.

[17] Budde, R. and Zullighoven, H. Prototyping revisited. *CompEuro '90. Proceedings of the 1990 IEEE International Conference on Computer Systems and Software Engineering,* 1990, p. 418–427.

[18] Davis, A. M. Operational prototyping: a new development approach. *IEEE Software,* 1992, vol. 9, iss. 5, p. 70–78.

[19] Luqi. Computer aided system prototyping. *1992 International Workshop on Rapid System Prototyping. Shortening the Path from Specification to Prototype,* 1992, p. 50–57.

[20] IEEE 830-1998. IEEE recommended practice for software requirements specifications. IEEE Computer Society, 1998.

[21] Royce, W. W. Managing the development of large software systems. *Proceedings, IEEE Wescon,* 1970, p. 1–9.

[22] Suganya, G. and Mary, S. A. S. A. Progression towards agility: a comprehensive survey. *2010 International Conference on Computing Communication and Networking Technologies (ICCCNT),* 2010, p. 1–5.

[23] TIOBE Software BV. TIOBE programming community index for February 2011. Online document, 2011. Referred 25.2.2011. Available at: `http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html`.

[24] Sabharwal, C. L. Java, Java, Java. *IEEE Potentials,* 1998, vol. 17, iss. 3, p. 33–37.

[25] Lewis, T. Bringing up Java. *IEEE Internet Computing,* 1997, vol. 1, iss. 4, p. 110–112.

[26] Kernighan, B. W. and Ritchie, D. M. The C programming language. 2nd edition. Prentice Hall, 1988.

[27] Ryan, R. R. and Spiller, H. The C programming language and a C compiler. *IBM Systems Journal,* 1985, vol. 24, iss. 1, p. 37–48.

[28] Cusumano, M. A. The changing software business: moving from products to services. *Computer,* 2008, vol. 41, iss. 1, p. 20–27.

[29] Nordman, T. and Hamberg, C. Master's thesis kick-off meeting, 3.1.2011.

[30] Birnbaum, D. and Vine, M. Microsoft Excel VBA programming for the absolute beginner. 3rd edition. Boston, Thomson Course Technology, 2007.

[31] Nordman, T., Kauppi, J. et al. Prototype demonstration session, 25.3.2011.

Appendix A: Functional design of the prototype