Assessment and Usability Test of Company Specific Hardware Configuration Tool

by

Robert Sandberg

LIU-IDA/LITH-EX-A--09/004--SE

2009-02-13
Final Thesis

Assessment and Usability Test of Company Specific Hardware Configuration Tool

Robert Sandberg

Reg Nr: LIU-IDA/LITH-EX-A--09/004--SE
Linköping 2009

Supervisor: Kristian Sandahl
IDA, Linköping University, Sweden
Stefan Holmberg
Ericsson AB, Linköping, Sweden

Examiner: Kristian Sandahl
IDA, Linköping University, Sweden
Utvärdering och Användbarhetstest av Företagsspecifikt Konfigurationsverktyg för Hårdvara

This master’s thesis consists of two parts. The first part is a study where the possibility to use existing tools from the market is assessed. The second part describes the design, development and usability test of a web based hardware configuration tool. The usability test made in the design process, was made in order to remove common usability problems and to assess whether or not the design decisions made during the thesis were good. The usability test that was performed followed the think aloud method and gave me very useful information that helped me improve the tool.

The study that was made in the beginning of the thesis was done in order to see if there were any programs on the market which already did the required tasks or could be adjusted to do them. I could not find a satisfying alternative amongst the five alternatives investigated. This lead to the decision to design a new tool.

The main functions are the ability to save the current state the hardware is in, and at a later stage load this configuration back.

The tool was designed for the PRAN unit at Ericsson AB, in Linköping, Sweden.

Keywords
Usability, Configuration, Tool, Hardware, PRAN, Ericsson
Abstract

This master’s thesis consists of two parts. The first part is a study where the possibility to use existing tools from the market is assessed. The second part describes the design, development and usability test of a web based hardware configuration tool. The usability test made in the design process, was made in order to remove common usability problems and to assess whether or not the design decisions made during the thesis were good. The usability test that was performed followed the think aloud method and gave me very useful information that helped me improve the tool.

The study that was made in the beginning of the thesis was done in order to see if there were any programs on the market which already did the required tasks or could be adjusted to do them. I could not find a satisfying alternative amongst the five alternatives investigated. This lead to the decision to design a new tool. The main functions are the ability to save the current state the hardware is in, and at a later stage load this configuration back.

The tool was designed for the PRAN unit at Ericsson AB, in Linköping, Sweden.
Acknowledgments

I am grateful for the input and guidance I received from my examiner, Kristian Sandahl, throughout the weeks of this thesis.

I would also like to thank the staff at Ericsson PRAN. Especially Stefan Holmgberg, Love Thyresson, Trevor Neish, Jonas Tevemark and Alexander Sandström Krantz who gave me crucial support when I was implementing features of the tool.
Contents

1 Introduction 1
  1.1 Background .......................................................... 1
  1.2 The PRAN Problem ..................................................... 1
  1.3 Purpose ................................................................ 2
  1.4 Method ................................................................ 2
  1.5 Structure of the Document ............................................ 3
  1.6 Abbreviations ............................................................. 5

2 PRAN Background 7
  2.1 The Need for IP-based RAN ............................................ 7
  2.2 The Ericsson PRAN Solution .......................................... 8

3 Literature Study 11
  3.1 Network Theory .......................................................... 11
    3.1.1 The OSI Model ...................................................... 11
    3.1.2 Media Access Control .............................................. 14
    3.1.3 Simple Network Management Protocol ...................... 14
    3.1.4 Virtual Local Area Network ................................... 14
    3.1.5 Physical Topology ................................................... 15
    3.1.6 Logical Topology ..................................................... 15
  3.2 Commercial Applications ............................................. 16
    3.2.1 PacketTrap .......................................................... 16
    3.2.2 HP OpenView Network Node Manager ....................... 17
    3.2.3 IBM Tivoli Network Manager Software ....................... 17
  3.3 Open Source Applications ........................................... 17
    3.3.1 OpenNMS ............................................................ 17
    3.3.2 NetDisco ............................................................. 18
  3.4 Usability ................................................................. 18
    3.4.1 Background .......................................................... 18
    3.4.2 Definition of Usability ............................................ 19
    3.4.3 Types of Usability Tests ......................................... 23
    3.4.4 Most Significant Usability Problems ......................... 26
Chapter 1

Introduction

1.1 Background

Ericsson has developed the Packet Radio Access Network (PRAN) solution to address the needs of operators that are moving to IP-based transport for Global System for Mobile communications (GSM) and/or Wideband Code Division Multiple Access (WCDMA) networks. The PRAN solution recommends equipment for the Base Station Controller / Radio Network Controller (BSC/RNC) and Radio Base Station (RBS) sites for use in an Internet Protocol (IP) Radio Access Network (RAN) environment. The reasons for moving to IP-based transport are the ability to handle increased amounts of data, more development possibilities and enable use of existing network architecture. A set of network scenarios are described; the setup of this environment is pre-configured and pre-tested to minimize additional network design and hardware configuration requirements at customer site. Aspects such as quality-of-service, traffic classification and prioritization in the IP RAN are highlighted in the design, together with security considerations including perimeter protection. The PRAN design can be adapted to fit an existing or planned IP infrastructure based on Ericsson or other vendors’ network equipment.

1.2 The PRAN Problem

PRAN provides operators with recommendations about the type of hardware that is suitable to use at the BSC/RNC sites. Hardware at these sites are for example Site Integration Units (SIUs), Security Gateways (SEGws) and Radio Access Network (RAN) switches. These come from different vendors, have different features and require different types of settings. These shall furthermore be connected to each other according to different models and the different combinations of connections demand different types of settings. In addition to this, each customer have demands of their own that sets more requirements on the configuration. To be able to provide a stable and secure solution, each case must be tested and verified.
This is time consuming. The hardware configurations have to be saved, not only for the final solution but also for each test case during development. Currently this is done manually and help is needed to improve and automate the configuration.

1.3 Purpose

The purpose of this thesis is to investigate whether or not a tool can be used to reduce lead times and costs concerning configuration of the above mentioned reference solutions. Commercial and Open Source tools that already exist on the market shall be evaluated to see if they can be used from the box or be adapted for use in this assignment. If no applicable tools exist on the market a proof of concept prototype shall be developed. The tool that will perform the required tasks shall be evaluated in the aspect of usability and necessity.

1.4 Method

During the initial phase of the project a literature study was done. There were two purposes of this study. The first was to provide the author with the proper knowledge to understand the task at hand. The second was to be able to determine whether or not to adapt an existing tool from the market or design and implement a new tool. As it turned out, a new tool had to be implemented.

The design and implementation of the new tool was done according to the waterfall principle \[2\]. This is normally divided into the steps analysis, design, implementation and testing. There are some situations that make this principle more suitable than other. These are listed below and all of which are applicable for this thesis.

- Many factors are known and/or are not possible to affect
- The company is strictly formal
- Development has well defined boundaries, for instance making of a prototype

The first step was to set up a meeting with the supervisor as well as with staff from Ericsson PRAN. The idea with this session was to merge the ideas from the supervisor with the ideas and demands the staff had. It was important to have this discussion since it is the staff that will use the tool. It is thereby of utmost importance to ensure that the tool would be of as much use as possible. The meeting resulted in an informal requirement specification. This is presented in Table 1.1 on page 3.

The second step was to present a design suggestion based on the requirement specification. This is an important step that will reduce the amount of double work due to misunderstandings. Some additional features were also brought up, considered useful and thereby added to the requirement specification.

The third step was the actual implementation. One of the requirements that were stated from the beginning of the thesis was that the tool was supposed to be
### 1.5 Structure of the Document

<table>
<thead>
<tr>
<th>Description</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>The system shall generate a configuration document based on the nodes in the network.</td>
<td>1</td>
</tr>
<tr>
<td>The system shall configure network equipment based on the above mentioned configuration document.</td>
<td>1</td>
</tr>
<tr>
<td>The system shall be expandable.</td>
<td>1</td>
</tr>
<tr>
<td>The system shall be graphical.</td>
<td>1</td>
</tr>
<tr>
<td>The system shall support the standard types of nodes that PRAN sells reference solutions for.</td>
<td>1</td>
</tr>
<tr>
<td>The system shall present uptime and downtime for nodes.</td>
<td>2</td>
</tr>
<tr>
<td>The system shall present and save firmware information.</td>
<td>2</td>
</tr>
<tr>
<td>The system shall generate a logical topology view based on layer 2 information of the network.</td>
<td>2</td>
</tr>
<tr>
<td>The system shall support edit functionality in the logical topology view.</td>
<td>2</td>
</tr>
<tr>
<td>The system shall generate a configuration document based on the logical topology view.</td>
<td>2</td>
</tr>
</tbody>
</table>

**Table 1.1. Requirements**

After this introduction a bit of Ericsson PRAN background is presented. The reader will get a brief introduction to what Ericsson PRAN is and will get further acquainted with the complexity of the system. In Chapter 3 on page 11 there are some basic network information followed by a study of available tools, which might have been of use for this assignment. To conclude the tool section of the report, Chapter 4 on page 31 describes the tool designed for this assignment.

The next major part of the report is the explanation of usability engineering. The theoretical part is presented in Section 3.4 on page 18. It starts out with describing what usability tests are and common usability problems. The usability test made in this thesis is presented in Chapter 5 on page 39.

At the end of the report, Chapter 6 on page 45, there is a discussion around platform independent. This narrowed down the possible programming approaches. After some careful consideration I decided to design a web based tool implemented in a scripting language. I discuss this in Section 6.5 on page 48. I was provided with a server and login information to a router and then I was on my way.

The forth step was to test and verify the prototype together with at least one of the end users. After designing a new tool or feature it is very good to do a usability test. Among other things, this is done in order to see the relevance of the tool, if the features are good or not, and most importantly, to find the most common usability problems. A usability test with the *Thinking-Aloud-Method* was performed. The result from this is described in Section 5.5 on page 42.
the different applications that were investigated and a discussion of the result from the usability test. The report finishes off with conclusions in Chapter 7 on page 53 for the entire thesis.
### 1.6 Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2G</td>
<td>Second Generation</td>
</tr>
<tr>
<td>3G</td>
<td>Third Generation</td>
</tr>
<tr>
<td>ASE</td>
<td>Application Service Elements</td>
</tr>
<tr>
<td>ATM</td>
<td>Asynchronous Transfer Mode</td>
</tr>
<tr>
<td>BSC</td>
<td>Base Station Controller</td>
</tr>
<tr>
<td>CDP</td>
<td>Cisco Discovery Protocol</td>
</tr>
<tr>
<td>CLI</td>
<td>Command Line Interface</td>
</tr>
<tr>
<td>CSMA/CD</td>
<td>Carrier Sense Multiple Access / Collision Detection</td>
</tr>
<tr>
<td>DNS</td>
<td>Domain Name System</td>
</tr>
<tr>
<td>FTP</td>
<td>File Transfer Protocol</td>
</tr>
<tr>
<td>GSM</td>
<td>Global System for Mobile communications</td>
</tr>
<tr>
<td>HSPA</td>
<td>High Speed Packet Access</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hypertext Transfer Protocol</td>
</tr>
<tr>
<td>IETF</td>
<td>Internet Engineering Task Force</td>
</tr>
<tr>
<td>IP</td>
<td>Internet Protocol</td>
</tr>
<tr>
<td>JMX</td>
<td>Java Management Extensions</td>
</tr>
<tr>
<td>LAN</td>
<td>Local Area Network</td>
</tr>
<tr>
<td>MAC</td>
<td>Media Access Control</td>
</tr>
<tr>
<td>MPLS</td>
<td>Multi-Protocol Label Switching</td>
</tr>
<tr>
<td>MS-DOS</td>
<td>Microsoft Disk Operating System</td>
</tr>
<tr>
<td>NSClient</td>
<td>Netsaint Windows Client</td>
</tr>
<tr>
<td>OSI</td>
<td>Open Systems Interconnection</td>
</tr>
<tr>
<td>PRAN</td>
<td>Packet RAN</td>
</tr>
<tr>
<td>RAN</td>
<td>Radio Access Network</td>
</tr>
<tr>
<td>RBS</td>
<td>Radio Base Station</td>
</tr>
<tr>
<td>RNC</td>
<td>Radio Network Controller</td>
</tr>
<tr>
<td>SEGw</td>
<td>Security Gateways</td>
</tr>
<tr>
<td>SFTP</td>
<td>Secure FTP</td>
</tr>
<tr>
<td>SIU</td>
<td>Site Integration Unit</td>
</tr>
<tr>
<td>SNMP</td>
<td>Simple Network Management Protocol</td>
</tr>
<tr>
<td>SSH</td>
<td>Secure Shell</td>
</tr>
<tr>
<td>TCP</td>
<td>Transmission Control Protocol</td>
</tr>
<tr>
<td>TDM</td>
<td>Time-Division Multiplexing</td>
</tr>
<tr>
<td>UDP</td>
<td>User Datagram Protocol</td>
</tr>
<tr>
<td>USB</td>
<td>Universal Serial Bus</td>
</tr>
<tr>
<td>VLAN</td>
<td>Virtual LAN</td>
</tr>
<tr>
<td>WCDMA</td>
<td>Wideband Code Division Multiple Access</td>
</tr>
</tbody>
</table>
Chapter 2

PRAN Background

2.1 The Need for IP-based RAN

Dedicated TDM lines have traditionally been used to transport traffic on the Abis and Iub interfaces between the radio base stations and the BSCs or RNCs [6]. Even though TDM is a reliable and established technology, new services for the end users such as HSPA and mobile TV are leading to increased demands on the network capacity and the increasing operating costs for TDM lines. A further challenge is, by 2011 the new data services desired by consumers may represent 90% of traffic load in the network, but only 30% of the operators’ revenue. In response to this, operators must completely re-evaluate their traditional networks to provide the necessary transport infrastructure and strive for the most cost-effective, high-quality solutions to succeed in the market.

The traffic growth in 3G networks can be derived from users who are normally connected to 2G networks but now migrate to the high speed 3G networks. However, new traffic due to the advantages of HSPA is the main growth factor and this is expected to have a massive effect on traffic volumes in the future. The increase in these traffic volumes are expected to come mostly from laptop users, equipped with USB modems, who only pay a fixed amount a month regardless of how much data they transfer, so called flatrate plans. Today, the number of users who have these types of solutions are relatively few and the traffic volumes they represent are relatively low. Many people feel that the reason why they do not use their mobile phones or laptops with HSPA modems for internet purposes is that the transfer rates are too low. But as the technology gets better, through Enhanced HSPA and Long Term Evolution, more people will go online and this will have a further impact on the increase of data volumes in mobile networks, requiring higher throughput and capacity.

Until now, mobile networks have not been required to transport large volumes of data, relying instead on the old voice oriented lines. For mobile operators with ambitions to compete in the Internet services market, dependence on the TDM lines to transport the increased amount of data, is financially unsustainable and a new solution is required.
Ericsson is now developing solutions, which will enable a transition from the use of dedicated lines to on-demand packet transport [6]. IP-based RAN takes advantage of already available IP/Ethernet infrastructure to transport user traffic, as well as different types of network maintenance traffic. The result is a solution which is easily scalable to support the increasing demands of data transfer and reduce the total costs for the operators due to lower transmission costs.

2.2 The Ericsson PRAN Solution

The Ericsson design provides a unified IP/Ethernet site infrastructure solution for GSM and WCDMA RAN in order to reduce the Total Cost of Ownership of the complete solution [6]. The PRAN solution recommends a variety of equipment and configurations for the BSC/RNC and RBS sites for use in an IP RAN environment. This enables operators to choose equipment and transmission technologies to suit their individual requirements when using IP infrastructure. When making the fundamental change to IP-based RAN, the operator must have absolute confidence in the design, implementation and maintenance arrangements for the new access network. Designing, implementing and maintaining the new network is a complex task demanding specific skills. But making these types of transitions are Ericsson’s everyday business. The reference solution is pre-configured and pre-tested to minimize additional network design and hardware configuration requirements. See Figure 2.1 on page 9 for an overview of the reference solution.

PRAN can be adapted to fit an existing or planned IP infrastructure based on Ericsson’s or other vendors’ equipment. It is designed to interact with Ericsson’s adjacent network solutions but those are not prerequisites for the PRAN implementation.
2.2 The Ericsson PRAN Solution

Figure 2.1. Overview of PRAN Solution
Chapter 3

Literature Study

3.1 Network Theory

3.1.1 The OSI Model

The OSI model, or more accurately the Open System Interconnection Basic Reference Model, was developed by ISO (the International Organization for Standardization) in 1984 and is now considered the primary architectural model for communication between computers [3], [4], [8]. It is a model that does not only describe the communication between computers but also the computer network protocol design. The OSI model shows how packets of data are moved in a network from software applications in one computer to software in another computer.

The OSI model can functionally be divided into of seven layers and two planes. The layers give an OSI system a vertical functional structuring and the planes provide horizontal structuring.

Because of the layer structure the OSI model is sometimes referred to as the OSI seven layer model. The level of abstraction increases as the layers numbers increases, see Figure 3.1 on page 12. A lower number symbolizes a more concrete layer. The seven layers structure a computer network from the transmission medium, such as cables or air, to the final network application.

While transporting data, from one computer to another, each layer only communicates with the layer directly above and below itself.

Layer 7 - Application Layer

This layer is the layer that handles the software used, which invoke the data transfer [8], [4]. The application layer is the highest level of abstraction and the highest layer of the OSI architecture. It is the layer that is closest to the user. The user uses software applications that have some sort of communicating component and this layer interacts with those. The actual programs are not part of the OSI model. Typical functions that the application layer handles are identifying communication partners, determining resource availability and synchronizing communications.
When an application has data to transmit the application layer determines the identity and availability of communication partners. The data that shall be sent need network resources and the application layer determines whether or not these exist. Software applications that communicate synchronized with each other have to cooperate and it is the application layer that handles this as well.

**Layer 6 - Presentation Layer**

If applications use different data representations the presentation layer ensures that the applications can communicate [8]. UNIX and Windows have different semantics and syntax and the presentation layer defines what happens when for instance UNIX data will be displayed on an MS-DOS screen.

**Layer 5 - Session Layer**

The layer that handles the connections between sessions is layer five [8], [4]. It establishes, manages and terminates those sessions. Service requests and service responses between different software applications on different computers in a network are the communication sessions. The protocols in this layer handles the
actual requests and responses. Commonly known protocols in this layer are the Zone Information Protocol (ZIP) and the Session Control Protocol (SCP).

Layer 4 - Transport Layer

The session layer sends data to the transport layer and this data gets segmented here [8], [4]. The transport layer ensures that the data gets to where it is heading, error-free and in the correct order. If a packet is lost (dropped), layer 4 notifies the sender and requests a new packet to be sent. This layer ensures that the three layers below, the network, the data link and the physical layer are doing their jobs. Thereby layer 4 provides reliable data transfer services to the upper layers. In case problems do occur, software belonging to layer 4 can handle the errors by for instance requesting packages to be resent.

It is also normally in the transport layer flow control occur. It means that the transmitting device does not send more data than the receiving device can handle.

Even though they were not developed as a part of the OSI reference model and do not strictly follow the definition of this layer, the best known examples of protocols that are used on the Internet and fall under layer 4 are TCP (Transmission Control Protocol) and UDP (User Datagram Protocol).

Layer 3 - Network Layer

The network address is defined by the network layer [4]. Some definitions of network addresses are defined in a way that makes it possible for routers to use this layer in order to determine how to forward packets. It is determined by systematically comparing the network address of the source with the network address of the destination and then applying the subnet mask. One of these network layer implementations is the Internet Protocol (IP).

Layer 2 - Data Link Layer

Procedures that enable nodes in a network to establish, maintain and release logical links belong to the data link layer [8], [4]. These links are used to transfer data units instead of raw bits. Different network and protocol characteristics such as physical addressing and network topology are defined by the data link layer. The data link layer also contains procedures and functions to monitor the integrity of the physical layer, perform bit errors detection and framing (delimiting streams of bits to form identifiable data units).

Layer 1 - Physical Layer

This is the least abstract layer [8]. Items belonging to layer 1 are routers, hubs, network adapters, copper wires, optical fibers, air (the case of wireless network), and etcetera. It contains the actual carriers of the data packets. A physical layer specification defines the transmission medium, the signaling technique, and the encoding scheme.
Planes
In addition to the seven layers, the OSI model is also divided into two planes:

1. Management plane
2. Operational plane

Functions for layer and system management belong to the management plane, [3]. The functional areas, which the OSI management takes into consideration are fault management, accounting management, configuration and name management, performance management and security management. The operational plane contains the communication functionality.

3.1.2 Media Access Control
All hardware, which operates on a network, has a unique fingerprint burned into a chip on it. This is commonly known as the Media Access Control (MAC) address. The hardware-based MAC address is packed into the data packets and then interpreted into the logical addresses which are used by layer 3 protocols.

3.1.3 Simple Network Management Protocol
Simple Network Management Protocol (SNMP) is described, in short, as a protocol that allows network administrators to connect to, manage and get alerts from network systems and devices [3], [5]. It is an application layer protocol that is used to exchange management information between network devices and is a part of the TCP/IP protocol suite. The SNMP is defined, designed and maintained by the Internet Engineering Task Force (IETF).

The typical SNMP use scenario is where a network has one or many systems or devices to be managed and one or many systems that act as managers. These managed systems or devices, sometimes called network elements, are typically routers, switches, hubs, computers or printers. Each managed system has an agent installed that sends information via SNMP to the managing systems. There are a number of system variables that usually are sent, such as free processor power, free memory, processes that are currently running and so forth. The managing systems can get this information in two ways. Either by requesting it by sending commands to the managed system or by having the agent sending the information due to so called traps. It is also possible to reconfigure the managed systems from the manager.

3.1.4 Virtual Local Area Network
A Virtual Local Area Network (VLAN) is a group of computers, routers and etcetera, which are not necessarily physically connected to each other but communicating as if they were connected to the same broadcast domain. A LAN and VLAN have the same attributes and have the common set of requirements but the hardware does not have to be connected through the same network switch.
3.1 Network Theory

3.1.5 Physical Topology

How the systems and devices are physically connected in a network is described by the physical topology [14]. The physical topology basically describes which wire is connected to which port in a certain device.

When designing a network it is important to choose the physical topology that is of most use to the network at hand. There are essentially five main types of physical topologies and all have different strengths and weaknesses. The five most common are:

- Bus
- Ring
- Star
- Hybrid or tree
- Mesh

3.1.6 Logical Topology

The logical topology describes how the systems and devices communicate over the physical topologies [14], [9]. There are essentially two main types of logical topologies. These are:

- Shared media topology
- Token based topology

Shared Media Topology

All the systems and devices in a shared media topology have the ability to access the physical layout whenever they need it [14]. The main advantage with this is that all systems have unrestricted access to the physical media. There is a disadvantage with this though. There is a risk for collisions and it increases when a larger amount of devices is connected to the network. A collision occurs when packets that the devices send out are transmitted on the same wire at the same time. A typical example of a shared media topology is Ethernet. A common way to decrease the number of collisions is to divide a large network into several smaller networks using switches and hubs. Normally the physical layout is carried out in a bus, star or hybrid physical topology.

In the Ethernet case, a protocol called CSMA/CD (Carrier Sense Multiple Access/Collision Detection) is used to avoid collision. In short, this works as follows. A system that wants to transmit data monitors and listens on the physical media for traffic. In case there is data on the media no transmission is initiated. When the wire is free, the system sends its packet. In case a collision occurs in spite of this the system waits a random time before sending its packet again.
Token Based Topology

For a device to get access to the physical media, a token is used in a token based topology [14]. There is one token available out on the wire when no one is sending data packets. When a system wants to send out data, it grabs the token off the wire, attaches it to the packets it wants to transmit, makes the transfer then puts the token back out on the wire. The token that follows the data packets carries information about where it came from (among other things). When the packets arrive at the receiving system, the system analyzes the information of the token and sends the token back to the sender. This works as a receipt. When the sender receives the token a new empty token is put out on the wire which the next device can use. The problem with collisions does not occur in a token based topology. But a disadvantage that does occur with this type of network is latency. Since the systems has to wait for the token before being allowed to send data a delay often occur when data normally could be sent.

3.2 Commercial Applications

3.2.1 PacketTrap

PacketTrap pt360 PRO is a network management application [7]. It is a software which combines several types of network analysis, network monitoring and network management tools.

PacketTrap have many normal functions common for most network management tools. Among those are functions for trace routing, sending ping requests and getting whois reports. In addition to these standard functions PacketTrap also provides monitoring functions of the nodes and some hardware configuration.

Information available through the monitoring functions is for example CPU use, memory use, disk use and average packet loss.

The hardware configuration PacketTrap support is for Cisco devices. It is possible to:

- Download device configuration files from Cisco devices
- Archive Cisco router startup and running network configurations
- Upload configuration changes to routers or switches via SNMP or Telnet/SSH
- Compare the running configuration of a Cisco (registered trademark) router with the startup configuration or archive configuration
- Go to and find any section within the configuration file quickly

PacketTrap provides layer 1 information, a textual physical topology, through the function Switch Port Mapper. It is possible to get information about all devices connected to each port on a switch and then identify each device by MAC address, IP address and host name.
PacketTrap also provides layer 2 and 3 information. A SNMP scan discovers the contents of network subnets by combining SNMP discovery capabilities with a ping scan of a designated range of IP addresses. Used and unused IP addresses are all identified and logged by SNMP scan as well. PacketTrap MAC scan sweeps the immediate subnet of its host and builds a table comprised of a pertinent MAC address, ping response time, DNS and network card manufacturer for each IP Address.

3.2.2 HP OpenView Network Node Manager

Network Node Manager is an industry-standard graphical SNMP network management application that provides fault, configuration and performance management for distributed multi vendor TCP/IP networks [10]. One of its key features is to perform automated discovery, topology mapping and monitoring of networks. Network Node Manager scans your network automatically and stores the discovered network information. This information can then be used to present the network graphically.

The application can discover any IP device. It can be computers, network elements (routers, switches, hubs), network based devices (network printers, file servers and etcetera) and display these devices in graphical form (maps/sub maps).

3.2.3 IBM Tivoli Network Manager Software

The IBM Tivoli Network Manager Software provides broad device and protocol coverage, it automatically discovers IP networks and network devices and it gathers and maps topology data and visualizes the layer 2 and layer 3 devices in a picture [18].

The physical layer information that is provided is given as the port-to-port connectivity between devices.

The logical layer information that is collected includes information about VPN (virtual private networks), VLAN (virtual local area network), ATM (asynchronous transfer mode), frame relay and MPLS (multi-protocol label switching) services.

3.3 Open Source Applications

3.3.1 OpenNMS

OpenNMS is a network management platform developed under the open source model and is mainly written in Java [11]. The OpenNMS application has a few main areas which it focuses on. These are:

- Service Polling - determining service availability and reporting on same
- Data Collection - collecting, storing and reporting on network information as well as generating thresholds
• Event and Notification Management - receiving events, both internal and external, and using those events to feed a robust notification system, including escalation

OpenNMS provides very detailed fault management. The faults OpenNMS detects are detected via three distinct and separate mechanisms:

• Service polling

• Receipt of messages which has not been requested (typically SNMP traps)

• Thresholds evaluated against performance data

OpenNMS also provides very detailed performance management. This is done via mechanisms that are based on a robust data gathering API called the Service Collector Interface. Current implementations of the Service Collector, SNMP, JMX, HTTP and NSClient, collect information that can be used in performance graphs and threshold analysis.

3.3.2 NetDisco

NetDisco is a web-based network management tool [1]. The information and connection data for the network devices and the topology information are fetched by issuing SNMP polls and DNS queries. There is a possibility to automatically discover the network topology if layer 2 topology protocols, such as CDP (Cisco Discovery Protocol), are used. One of its features is that it can locate the switch port of an end-user system by MAC address or IP address.

3.4 Usability

3.4.1 Background

For every web designer, who creates a web page that has a purpose for the user and if the user is trying to accomplish something on the site, usability is important [15].

Usability is a quality attribute measuring how easy something is to use [12]. If people can learn how to use a tool easily, if they are efficient while using it, if it is not error-prone and if users like the tool, then it is a tool with good usability.

Nielsen says that when performing usability testing the test persons should be carefully selected. It is a good idea to for example screen out anybody working in technology, marketing, web design or usability because they rarely represent mainstream users, if the system that will be evaluated is a normal web page. Those who work in those areas criticizes the design based on their own personal design philosophy and may have difficulty engaging with the design as intended. The main idea is to test usability with the type of persons who will use the system.

When it comes to site-specific testing the way most usability studies are conducted is telling users where to go and that they are expected to stay there while
performing their tasks. The most preferred approach for almost all usability tests is, according to Nielsen, the "thinking aloud" method, which implies that the users are asked to think out loud as they work with the interface. This can be combined with two cameras, one recording the users face and upper body and one recording the computer screen. The really valuable information is not only to know that a user clicked the wrong button, even if that also is good to know, but to know why the user clicked the wrong button.

A usability test’s major result will be a list of the usability problems in the interface as well as hints for features to support successful user strategies. It is normally not feasible to solve all the problems so one must prioritize.

3.4.2 Definition of Usability

Having an interface with good usability is not determined by a single attribute. It is rather a combination of several usability attributes, normally divided into these five sections:

- Learnability
- Efficiency of use
- Memorability
- Errors
- Satisfaction

These sections will be discussed further in the following sections. Dividing usability into these areas is a good aid in order to perform good usability tests. The areas stated above are areas which more easily can be broken down into more precise and measurable components.

Normally when a usability test is performed a number of test users use the system after they have been given prespecified tasks [15]. It can also be performed by users doing their normal tasks they always do. Either way, the important part is to have the test constructed to suit specific users for specific tasks. Different types of users may not use a system the same way since their tasks are different. For example, someone writing a letter to a pen pal in Microsoft Word does not use features the same way as a technical writer do when writing code documentation.

Learnability

This is perhaps the most important section where the usability needs to be good [15]. If a system is easy to learn it will be used a lot more than if the system is hard to learn. The first experience users have with a new system is learning how to use it and if it is not a good experience then there will be certain resilience toward the system. If advanced tasks are supported by the system then it is of course sometimes necessary to have a complex system. But in any case systems need to be easy to learn.
When designing new systems sacrifices are normally done. It is commonly necessary to determine what the systems complexity should be. This will in a sense determine who will use it. Systems designed for novice users might be easy to learn, but an easy-to-learn system is more seldom as efficient and can perform the same difficult tasks as a more complex system. This is illustrated by the learning curves in Figure 3.2 on page 20.

![Figure 3.2. Learning curves for a hypothetical system that focuses on the novice user, being easy to learn but less efficient to use, as well as one that is hard to learn but highly efficient for expert users.](image)

How easy a system is to learn is perhaps the easiest usability attribute to measure. The experimenter simply picks a number of users who have never used the system before and measure the time it takes for the users to reach a certain level of expertise. It is of course necessary to pick users who represent the intended users of the system. It might seem difficult to express what the specified level of expertise is, but it really is not. Simply stating that a user needs to be able to complete a certain task is something that is measurable.

**Efficiency of Use**

Efficiency is normally referred to as the expert user’s steady-state level of performance at the time when the learning curve flattens out, see Figure 3.2 on page 20. This is not a state users get to particularly fast and some systems are so complex that it takes years to reach that level of understanding [15]. And even if a user manages to get to this level, it might not always be worth the effort. The time it took to learn the advanced features might not always pay off.

To be able to measure efficiency one needs to get in touch with experienced users. It is not always easy to define which users are experts and which are not. It is normally decided informally and it is common that the users themselves claim to be experts. Another way of measuring experience is by checking the number of hours spent using the system. This way is common when it comes to
experiments with new systems without an established user base. In this case test users are brought in and asked to use the system for a certain number of hours. Their efficiency is later measured. Another way to define a user’s experience is by comparing the user’s performance in comparison to the learning curve. The test user’s performance is measured and when the performance has not been increasing for some time the steady-state level of performance for that user has been reached.

After having selected suitable test users, a normal way of measuring efficiency is by measuring the time it takes these users to perform some specific test tasks.

Memorability

There are three major types of users [15]. There are novice users, experts and of course the causal users who are in between. Casual users do not use the system as frequent as the experts but they have used the system before, in contrast to the novice users. Since they do not have to learn the system completely from the beginning all they have to do is to remember the way they used it the previous times. Casual use is typically seen for programs that are used under certain circumstances, additional or aiding programs to a user’s primary work and for programs that are used at long intervals. This could be programs for making a quarterly report.

An interface that is easy to remember is not only important for those scenarios. Users that have been on vacation or for some reason have not been working with the program for some time must be able to quickly get back on track. If a system has good learnability it is quite common that the memorability of the system also is good. But this is not always the case. Nielsen wrote a nice example about this. He mentioned that there is a sign with the text “Kiss and Ride” written on it which can be seen outside some Metro stations in the U.S. The meaning of this sign is perhaps not obvious at first, it has poor learnability without outside help. But once you realize what it means, it is easy to remember. A “Kiss and Ride” area is a place where people who commute drop of passengers who will continue their way to work by train. The “Kiss” refers to commuters who are driven by their spouse and kissed good bye.

Memorability is rarely tested as thoroughly as the other areas of usability. There is normally two ways you measure memorability. The first is to have casual users, who have been away from the system for a specified time, perform some typical test tasks. In that case you measure the time it takes for the user to perform the task. The second way to measure memorability is by letting a user answer questions after having used the system. The questions are performed as a memory test and the score of the test is the number of correct answers given by the user.

The first way of measuring is the most representative way we want to measure memorability. The problem is to find suitable test users and to analyze the result. First you have to find test users who are in the category casual users and that they have to have been away from the system for some time. Secondly, the time it takes to perform a task is individual. Different timing for different users can be hard to analyze, some work faster than others. The memory test is much easier
to perform and to analyze, either the test users answer the questions correct or not. But this does not always reflect the system in a fair way. A user might not remember all the options in a menu after having used the system, but they might not have any problems using it even after they have been away from the system for some time. The interface itself might be designed to help the user remember for example clickable buttons.

Errors

One could say that there are two types of errors [15]. There are the severe errors that might compromise the quality of a product or damage something for the user making it difficult to recover from. There are also the minor errors that only slow down the user. The minor errors are often mistakes that the users notice and correct immediately.

Since both types of errors prevent the users to reaching their desired goal they are best to get rid of. Normally one does not have special tasks designed for measuring errors in usability tests but rather counting errors continuously when performing other test cases. When counting, it is important to separate the severe errors from the minor ones. The most crucial errors should be solved first.

It is good to define what a severe problem is. Problems with high severity are problems that lead to unacceptable costs or lost business. Problems with medium severity are problems which cause confusion and frustration. This might result in users not using the system which cause lost business. Low severity problems are normally only cosmetic or irritating and will individually not affect the use of the system. Although, many minor problems can lower the total experience of the system which can result in lack of use.

Satisfaction

This usability attribute refers to how pleasant it is to use the program [15]. Depending on which system you are evaluating this attribute has different significance. If you for example have a system designed for a non-work environment, such as a computer game, then you might want to spend a lot of time using the system.

Nielsen describes a problem, and as a quite big one when it comes to measuring subjective satisfaction. He mentions that there might be resistance towards computers in general which of course impact the way a user feels about the system being evaluated. His reasoning was more applicable when the book was written. In today’s society this resistance has decreased. Even though this problem still exists I must believe this has changed in a way that people are positive toward computers in general nowadays.

In a few cases psychophysiological measures such as EEGs, pupil dilation, heart rate, skin conductivity, blood pressure and level of adrenaline in the blood are used to estimate the user’s stress and comfort levels during a usability test. This is rarely suitable for usability engineering purposes since the test user normally should evaluate the systems in a normal and relaxed atmosphere. Test users are stressed and
nervous enough during testing and the psychophysiological measurement methods would be too much.

It is the general opinion about the system we are looking for. We want to know how satisfied the users are with the program. A good alternative is simply to ask them. Getting response from one user gives a subjective opinion of the system. When asking many users, one cannot take every opinion into consideration when it comes to redesigning a program. But the average result of the replies from all the users gives the system’s pleasantness. This procedure is done in most usability studies.

When asking these questions it might be a good idea to have some questions with so called reversed polarity. This means that an agreement to the statement or question would be a negative rating for the system. Having some questions like this partly counteract the politeness people have when rating a system. Users are normally too kind when asked for their opinion. They will tend to be a bit more positive unless they have had a really bad experience.

If several systems are available that is supposed to do the same tasks, users could be asked to tell you which system they prefer. They can also be asked to rate how strongly they would prefer one system over another. Finally, if a system is already released and users are aware of it one can measure how many users that use that system over others. The most best subjective satisfaction rating is when you have data showing voluntary use.

3.4.3 Types of Usability Tests

After you have read the brief information about usability attributes above you might realize that the area of usability testing is big [15]. There are many perspectives to take into consideration when testing the usability of a system. Since there are so much to test and as I have written above, some usability attributes can be tested in many different ways, all which have their advantages and disadvantages, it might not come as a chock when I say that there is also many different methods when it comes to testing. I will present a summary of the most common usability methods below, in Table 3.1 on page 29, and thereafter describe the method I chose for my usability test a bit more detailed.

If you look in Table 3.1 on page 29 you see that the different methods complement each other, not only in terms of advantages but also where in the design process the system is. It is of this reason a good idea to not only rely on one method but through the duration of the system development perform many usability tests. Table 3.1 on page 29 also shows that the number of test users that are available affect the types of methods that can be chosen. When fewer users are available heuristic evaluation, thinking aloud and observation methods are suitable. If more users are available you might rather choose performance measurement or focus groups and if a very large number of users are available then questionnaires, interaction logging and systematic collection of spontaneous user feedback can be considered.

According to Nielsen, the experience of the people who are performing the tests may also impact the type of method to choose. The more simple methods
are thinking aloud and observation since less is required from the usability person. The usability person does not have to do much more than be quiet during the actual test.

Combining Usability Methods

Since there are so many things to check for when it comes to usability there are many possible ways of combining the different types of methods [15], [13]. The same combination might be more or less successful for different projects, which mean that each case is special. But a combination that has proven to be useful in many cases is that of heuristic evaluation and thinking aloud. The heuristic evaluation is performed first in order to clean up the interface and remove the most obvious flaws. After this has been made and a possible redesign has taken place, user testing in the form of thinking aloud can take place. There are two major reasons for doing it in this order. The first is that you do not ’waste’ users, their time and patience, with usability problems that can be eliminated in advance. The second reason is that these two methods complement each other in the sense that they do not find the same usability problems.

As another example interviews and questionnaires can be combined. Using a small number of users to get into depth and analyze specific issues through open interviews and then later on using findings from the interviews to create valuable questions in closed questionnaires, which are sent out to a large number of users.

Heuristic Evaluation

Heuristic evaluation is essentially 'look and evaluate' [15], [13]. A number of so called evaluators look at the interface, just on a paper edition, as a prototype or in a full featured system, and come up with an opinion of what is good and bad. This is normally done having a set of guidelines as base at which they check if the interface follows a certain set of rules [12]. It has been shown by Nielsen and Molich that a single evaluator only will cover about 35% of the usability problems in a system. Figure 3.3 on page 25 shows the average results from six studies discussed by Nielsen. It is quite easy to see that having at least three evaluators substantially increases the number of problems found. Between three and five evaluators are recommended to have. Having more than five evaluators usually do not pay off. The cost increases more than the benefits do.

The evaluators can perform the test by themselves or with an experimenter present. In the latter case the evaluators will be assisted when they do not understand the system and is clearly in trouble, especially if it is a domain-specific system and the evaluators are not experts. This allows the evaluator to better access the interface as intended. The evaluators are not supposed to actually use the system. This is why they really do not have to know what they are doing or even to have a working system. Even a paper version in the beginning of the design phase can be very useful.

When the evaluators are doing the test they use the system several times. Every time checking options, dialogue elements and so forth and compares them to a list of recognized usability principles. The usability principles are a set of general
rules that has been proven useful to follow in order to have a usable interface. The evaluators may of course also take their own opinions into consideration, note this and present it to the experimenter.

Think Aloud

Again, according to Nielsen, when it comes to user testing the think aloud method is considered to be the single most useful usability engineering method [15], [13]. The test users are asked to think aloud as they perform a set of tasks using the system. This makes it possible for the experimenters to understand how the users interpret the different items in the interface and thereby understand why certain objects can cause problems.

A disadvantage with this method is that it does not feel natural to the user to say what he/she thinks while he/she performs certain tasks. While talking, the user might think through a decision an extra time before acting. These things affect the user performance, which is a big disadvantage since performance measurement become faulty. Major advantages is on the other hand the amount of data one can
collect, even only if you have few users who test the system. It can be everything from discovering big usability problems to small things that are really just irritants, but things that would never show up in other forms of testing. A real strength with this method is to present what the users are doing, why they are doing it, while they are doing it. This makes it easier to avoid later rationalizations.

Since it feels unnatural for most users to say their thoughts some users may have difficulties saying anything as they use the system. In order not to affect the users the experimenter has to be careful of what he/she says during the tests. It is good to prompt with questions such as "What are you thinking now?" or "What do you think this message means?" after the user clearly has seen the message of course. During think aloud tests the experimenter is not allowed to help the users. If the user asks a question such as "Can I do this-or-that?" the experimenter is not allowed to answer yes or no. A better way of handling that situation is to keep the user talking by asking a counter-question such as "What do you think happens if you do it?".

3.4.4 Most Significant Usability Problems

There are many common usability problems, more or less meaningful, but every designer should take as many as possible into consideration when designing a system [17]. I will briefly mention the top areas in this chapter.

Graphic Design and Color

If it is not a Command Line Interface (CLI) then graphic design and color is a very important aspect since all communication between computer and human go through the graphic interface [17], [12]. Things to keep in mind are for instance that elements on the screen that belong together should stay close to each other or be enclosed by lines or boxes. It is also possible to use colors to categorize, highlight information and etcetera. When it comes to web pages there are a few things that has been proven bad. These are:

- Links that do not change color when visited
- Disabling the feature to step back to a previous page
- Pop-up windows
- Design elements that look like advertisement
- Dense content and unscannable text
- Cross-platform and browser incompatibility

At a conference in 2004, John Boyd, Manager of Platform Research at Yahoo! and Christian Rohrer, Director of User Research at eBay presented result from a survey done on how users perceive online advertising [17]. 605 users participated in the survey and a part of how they experience online advertisement is shown in Table 3.2 on page 30. Even though there are rarely ads in company specific tools, it is a good idea to keep away from the same type of design tricks used by ads.
3.4 Usability

Less is More
Large amounts of information tend to scare users or at least confuse users [17], [15]. Keep the information brief and concise. This does not only apply to text but also to features. Every new feature that is added is a new feature the user has to learn. Additional features that are not necessary for every user to use should be kept in an 'Advanced' section.

Speak the User’s Language
Information should be presented in the user’s terminology and not in system oriented terms [15]. Also, when informing the users of say an action they have made, consider what has happened from their point of view. For instance, 'You have sold this or that to us' instead of 'We have bought this or that from you'.

Minimize User Memory Load
Help the user remember what to do and what is happening [15]. Make sure that the system for example tells the user what to fill out and how to fill out a form. Enable functions such as "auto-completeness" for commands or command listing functions.

Consistency
Make sure that the same feature always will give the same result [15]. This consistency gives the user confidence in the system. Keeping the graphical design the same on every page will help the user feel that he/she is on the right place.

Feedback
As internet connections get better there is the possibility that the time a user has to wait for each web page to load decreases [17]. This is not always the case since web pages in parallel to this get more and more complex. Slow download times is still a usability problem. For normal web pages on the Internet this is a big problem. Studies have shown that there are some time limits to keep in mind [16]. These are:

- A user will feel that the system is reacting instantaneously if his actions are presented within 0.1 seconds. In this case no special feedback is required.

- The user’s flow of thought remains uninterrupted if response is given within 1.0 seconds, but the delay is noticed. Normally no special feedback is required in this case either.

- If the delay is up to 10.0 seconds the user can still be focused on the dialogue but will be annoyed. When the delays get longer the user tends to do other tasks while waiting for the page to finish. Feedback is important in these cases.
When it comes to the use of tools the users normally have greater patience with delays since they know that some background tasks are being performed and that they are not only waiting for a web page to load. In any case feedback is essential. Otherwise the user might abort the task even though nothing is wrong.

**Clearly Marked Exits**

Make sure that the user has an exit from every state in the system [15], [17]. Preferably both back and forward. If the back option for some reason is not available an undo option should be available instead. A good thought to keep in mind is that no matter how explanatory a system is users will still make errors. Therefore one should always give the user the possibility to recover from these errors.

**Shortcuts**

Advanced users normally use keyboard shortcuts in the tools they use [15]. But it is also common among casual users to use shortcuts to some extent. Enabling this will make users produce result faster. For example, the <TAB> key is commonly used in UNIX environment to speed up writing commands.

**Good Error Messages**

Error messages should be informing as well as instructing [15]. If a user gets an error message because he/she used a system incorrectly it is better to say "Wrong password given" instead of "Error code: 3123". If error messages are intended for users as well as to inform the system administrator both message types can be visible. It is also a possibility to have the full error message stored in a log file, which the administrator can read for himself.

**Prevent Errors**

Design the system with limitation in order to minimize the risk for errors [15]. Use for instance spell checks of different sorts to ensure that fields are filled out correctly. If the user has requested to an action that might give serious consequences ask the user to reconfirm that they "really, really mean this".

**Help and Documentation**

The most preferable system is a system that does not require additional documentation than the interface itself [15]. This goal can rarely be met but should be the tried to achieve for every system. Keep the documentation simple. Users seek out the documentation to be able to understand a feature or grasp a problem, not to get more confused. In case there are several levels of understanding a feature, divide the instructions in basic and advanced sections.
<table>
<thead>
<tr>
<th>Method Name</th>
<th>Lifecycle Stage</th>
<th>Users Needed</th>
<th>Main Advantage</th>
<th>Main Disadvantage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heuristic evaluation</td>
<td>Early design, 'inner cycle' of interactive design</td>
<td>None</td>
<td>Finds individual usability problems. Can address expert user issues.</td>
<td>Does not involve real users, so does not find 'surprises' relating to their needs.</td>
</tr>
<tr>
<td>Performance measures</td>
<td>Competitive analysis, final testing</td>
<td>At least 10</td>
<td>Hard numbers. Results easy to compare.</td>
<td>Does not find individual usability problems.</td>
</tr>
<tr>
<td>Thinking aloud</td>
<td>Iterative design, formative evaluation</td>
<td>3-5</td>
<td>Pinpoints user misconceptions. Cheap test.</td>
<td>Unnatural for users. Hard for expert users to verbalize</td>
</tr>
<tr>
<td>Observation</td>
<td>Task analysis, follow-up studies</td>
<td>3 or more</td>
<td>Ecological validity; reveals users’ real tasks. Suggests functions and features</td>
<td>Appointments hard to set up. No experimenter control.</td>
</tr>
<tr>
<td>Questionnaires</td>
<td>Task analysis, follow-up studies</td>
<td>At least 30</td>
<td>Finds subjective user preferences. Easy to repeat.</td>
<td>Pilot work needed</td>
</tr>
<tr>
<td>Interviews</td>
<td>Task analysis</td>
<td>5 or more</td>
<td>Flexible, indepth attitude and experience probing.</td>
<td>Time consuming. Hard to analyze and compare.</td>
</tr>
<tr>
<td>Focus groups</td>
<td>Task analysis, user involvement</td>
<td>6-9 per group</td>
<td>Spontaneous reactions and group dynamics.</td>
<td>Hard to analyze. Low validity.</td>
</tr>
<tr>
<td>Logging actual use</td>
<td>Final testing, follow-up studies</td>
<td>At least 20</td>
<td>Finds highly used (or unused) features. Can run continuously</td>
<td>Analysis programs needed for huge mass of data. Violation of users’ privacy.</td>
</tr>
<tr>
<td>User feedback</td>
<td>Follow-up studies</td>
<td>Hundreds</td>
<td>Tracks changes in user requirements and views.</td>
<td>Special organization needed to handle replies</td>
</tr>
</tbody>
</table>

Table 3.1. Summary of Usability Methods
<table>
<thead>
<tr>
<th>Design Element</th>
<th>Users Who Answered 'Very Negatively' or 'Negatively'</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pop-ups in front of your window</td>
<td>95%</td>
</tr>
<tr>
<td>Loads slowly</td>
<td>94%</td>
</tr>
<tr>
<td>Tries to trick you into clicking it</td>
<td>94%</td>
</tr>
<tr>
<td>Does not have a Close button</td>
<td>93%</td>
</tr>
<tr>
<td>Covers what you are trying to see</td>
<td>93%</td>
</tr>
<tr>
<td>Doesn’t say what it is for</td>
<td>92%</td>
</tr>
<tr>
<td>Moves content around</td>
<td>92%</td>
</tr>
<tr>
<td>Occupies most of the page</td>
<td>90%</td>
</tr>
<tr>
<td>Blinks on and off</td>
<td>87%</td>
</tr>
<tr>
<td>Floats across the screen</td>
<td>79%</td>
</tr>
<tr>
<td>Automatically plays sound</td>
<td>79%</td>
</tr>
</tbody>
</table>

Table 3.2. Most Hated Advertising Techniques
Chapter 4

PRAN Hardware Configuration Tool

4.1 General Information

PRAN Hardware Configuration Tool is the tool developed for this thesis. It is a web based tool mainly written in Perl and HTML. Data that is input to the tool and data that is collected from different hardware types is stored on a server and in a MySQL database.

It runs on a Linux server with Apache 2.2.9, Perl 5.8.8 and MySQL 5.0.45 installed.

4.2 Functionality

The tool is, as written above, a web page. The first page the user meets look like Figure 4.1 on page 33. There are three main functions presently implemented. These are:

- Saving configuration
- Presenting saved configurations
- Loading a saved configuration

Instead of saving configuration files one node at a time, this tool lets the user set up connections for all the nodes involved in the network at once. This enables the possibility to save the current setup of hardware as a group and all configuration files are saved at once. Furthermore the user does not have to log on to each router through SSH/Telnet but can simply enter a web page and fill out the information needed. Figure 4.2 on page 34 shows how the page looks where this information is set.
The save function connects to the nodes as background processes and send commands through an SSH client. The configuration files are then sent through SFTP from the nodes to the server. The time consuming parts are not the actual data transfers but the time it takes to confirm that correct return values has been received after sending commands. To save time the save function is implemented with parallel forking procedures, starting a new process for each connection. While this is happening in the background, the user sees a web page where progress information appears. Figure 4.3 on page 35 shows what this page looks like.

When a set of connections have been established and the configuration files have been successfully saved, data is written to the MySQL database in order to retrieve the saved data easily. At the time of writing this thesis the presentation function simply retrieve some of the data stored and present it on a web page, as shown in Figure 4.4 on page 35. A search function will hopefully be implemented later. See Section 5.5.2 on page 42 and Section 6.6.3 on page 50 to see a discussion around the search function.

On the presentation page a list of all the saved configuration files from the nodes is presented, as in Figure 4.4 on page 35. By selecting one of these and clicking a button, the user will be asked to enter the login information to the nodes, if he wishes to load the configuration files back into the nodes. This page looks like Figure 4.5 on page 36. At the same page, the user can click a link and view the contents of each configuration file. The content is presented as in Figure 4.6 on page 37.

### 4.3 Interface

The tool is web based, meaning it is used by entering a web page. It is structured as a form, with fields required to be filled out. The fields contain information such as IP address to the hardware, login name and login password.

Status messages and other information the user might want to see is generated on a web page as the processes progress.

### 4.4 Features for the Future

During the course of the thesis work, more desired functions were brought up. Some of the functions requested were:

- Diff - Possibility to compare two configuration files
- Support for more hardware types
- Search engine for presentation page
- Edit possibility for configuration files
Figure 4.1. Start page
Figure 4.2. Save page
4.4 Features for the Future

**Figure 4.3.** Progress page

**Figure 4.4.** Present page
Figure 4.5. Load page
4.4 Features for the Future

Figure 4.6. More information page
Chapter 5

Usability Test

5.1 Test Execution

As I wrote in Section 3.4.3 on page 24, it could be a smart idea to first do a heuristic evaluation followed by a user test. The heuristic evaluation was not done and the reason is presented more thoroughly in Section 6.6.1 on page 49. What I did instead, prior to the real user test, was to have a combination of a think aloud test and a walk-through of the tool with another test user. This was done in order for me to practice as an experimenter and at the same time get some user feedback about the tool. The result from this test is described in Section 5.4 on page 40.

For my usability study, I decided to do user tests with the think aloud method. The test users were given the same introduction where I explained the purpose of the tool and the purpose of the study. I explained what they could expect from me as an observer during the test and I explained the expectations I had on them. The expectation I primarily had on them was to think out loud, say their thoughts out loud. Afterwards, I handed out a paper with the tasks they were supposed to do written on it. The tasks they were given are presented in Appendix D on page 67. When the test users had read through the task description, I allowed the users to ask questions about the tasks in order to prevent any misinterpretations during testing.

When each user was finished with all the tasks, I had a debriefing session. During this debriefing, I asked a couple of additional questions about the tool. These are also found in Appendix D on page 67. After that, I asked if the user had any suggestions for improvement of the tool. The result from each test and the discussion I had with each test user is presented in Section 5.5 on page 42 and I discuss the result in Section 6.6.3 on page 50.
5.2 Test environment

The usability test was done in standard office environment with a laptop similar to the ones that the potential users have. The same type of external computer screen was used, the same internet connection and the same Internet browser was used.

5.3 Evaluators

One very important rule to have in mind, when selecting test users, is to make sure that the users are as representative as possible for the end users. This is most important if the number of test users are few. In the case of my study the individuals who will use the system are known. The tool is being developed for a certain department, which makes it easier to find test users. The difficulty lies instead in getting them to spend time with the testing instead of their primary jobs.

There were three users selected for this study. They are experts in the area for which the tool is designed and two of them are also end users. The first user described in Table 5.1 on page 40 is not an end user and does not have the full technical knowledge of the equipment behind the tool. He has got a more general view of the system. It was he who made the first experiment test, see Section 5.4 on page 40.

<table>
<thead>
<tr>
<th>Test User</th>
<th>Age</th>
<th>Education</th>
<th>Work Title</th>
<th>Sex</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>28</td>
<td>Computer Science and Engineering</td>
<td>Solutions Verification Engineer</td>
<td>M</td>
</tr>
<tr>
<td>2</td>
<td>24</td>
<td>Information Technology and Engineering</td>
<td>Solutions Verification Engineer</td>
<td>M</td>
</tr>
<tr>
<td>3</td>
<td>25</td>
<td>Computer Science and Engineering</td>
<td>Solutions Verification Engineer</td>
<td>M</td>
</tr>
</tbody>
</table>

Table 5.1. Evaluators

5.4 Experiment Test

5.4.1 During the Test

The result from the usability matrix is not presented since it was only filled out for my own personal use and for me to get some experience executing usability tests. What I learned about myself from this test is presented in Section 6.6.2 on page 49. What I can say about the tool is that some of the fields caused some minor confusion but since I was not quiet enough no errors where encountered. The issues that were brought up during the discussion after the test is presented in Section 5.4.2 on page 41. Right after the test user completed the final task he
said some words that summarize his opinion and feel for the tool. The test user said: "Simple to use in comparison to the ways of working now and it has no fields that are unnecessary. It does what it is supposed to do".

5.4.2 Debriefing Session

The design problems that were discovered during this initial test were not severe problems. They should be considered as irritants. When we discussed the tool after the tasks had been completed, design problems and feature problems where brought up. These are presented below.

Save Page

- Field order
- Password fields

It felt more natural if the order of a couple of fields were switched. The field that the user felt should come first was "Hardware type" and not "IP address", which was the case at time of the test. It felt more natural to select what type of hardware to save the configuration from first and then specify information about the hardware.

The password fields were at the time of the test designed as hidden text fields, meaning that characters entered in the fields only appeared as stars. This felt most natural to me as a designer, which is why I did it this way. The test user pointed out that this tool will only be used for internal purposes, it will not come into the "wrong hands", and that spelling mistakes could be avoided if the characters were visible. This would of course save time since the user would not have to wait for the tool to connect to each node, find out that the password is not correct and then go back and fill out the proper password.

Load Page

- Pre-filled out username and password fields

Out of security reasons the usernames and password was not saved into the database and filled out in advance at the load page. But with the same reasoning as above, the information for the login fields filled out during save could be stored in the database and filled out in advance at the load page.

More Information Page

- Edit configuration file

During this test, the only issue that was mentioned about this page was actually only a feature improvement. The test user pointed out that it would be a good feature to be able to edit the configuration file from the web interface.
5.5 Think Aloud Test

I sat next to the test users and filled out the usability matrix as they executed the four tasks. The clean matrix is found in Appendix A on page 61 and the filled out sheets are found in Appendix B on page 63 and in Appendix C on page 65. I will first summarize the result from the matrixes and then present what each test user had opinions about.

5.5.1 Summary of Matrixes

Both test users expressed pensiveness more often than they expressed satisfaction. But in both cases, there were more user statements that were positive towards the system than there were statements that were critical towards the system. Even though both of the test users where pensive, neither of them wanted more feedback or progress information from the tool and neither requested a help page during the session.

Both of the users solved their problems without getting stuck at a specific task. They did also find their way around the web pages but did it in very different ways. One of the users had a more structured way of working where he read the information and selected the appropriate links to continue through the tool. The other went more with the trial and error strategy. This lead to some backward-forward clicks in order for him to find his way.

As they navigated through the tool and used the existing features, both of the users came with improvement comments. During testing they mentioned a couple of things, but more was brought up during the debriefing session afterwards.

5.5.2 Debriefing Session

In the following section, I will bring up what was mentioned during the debriefing session. But I will only bring up the problems, errors, suggestions of design improvements and possible feature improvements. All that was positive towards the system will be summarized in Section 6.6.3 on page 50 and Chapter 7 on page 53.

Save Page

Problems and suggestions with the save page that were discussed afterwards:

- Clarify the purpose of the first field on the page
- Rename the headline for each connection
- Rename router to hardware
- Select hardware before entering other information
- Have password fields as normal text fields
- Clarify IP address to server from router
• Have each field on separate rows

The first and foremost problem on this page was the top field where the user is supposed to select the number of connections he or she wants to set up. Text explaining that it is the number of nodes one wishes to connect to would be more suitable. Information stating that this selection must be done first would also be good. The users did not expect to get such an empty page as it is when the number of connections is set to zero.

The headline for each set of connection settings is now for example Connection 1. The users felt that it was a bit confusing and a better headline would be something like Node 1 - Connection settings and comments

The tool is not only supposed to support routers in the future. This is the case at the moment, but it would be better to rename router to hardware. The tool is hopefully going to be developed further to also support switches and for example SIUs.

When one of the users was going to enter the IP address to the router he was going to save the configuration from, he said: "I guess this IP address is to the router". This uncertainty could have been avoided if he would have selected the hardware type before entering any connection information. The other test user also found it to be more natural to do it in that order, even though he did not say anything during the test.

As discovered during the experiment test, these test users also felt it to be unnecessary to hide the characters when entering the passwords. They also had the opinion that the tool would not fall into the wrong hands. To increase security, one of the users came with the idea that a single sign in, a login page, to the tool could be used and then have all fields visible and all information saved to the database and load page.

One of the testers was confused about the field where he was supposed to fill out the IP address to the server. Which server and what it has to do with this tool was of course not obvious. Text describing that it should be the IP address to the server, where the configuration files should be saved and where the database is, would make it easier to understand.

One user came with the suggestion that each field, which is supposed to be filled out should be on separate rows. The reasoning was, when a field was filled out, the row was completed, and it would give the feeling that one is ready to continue with the next part.

Submit Page

• Larger text or some form of notification that shows that the save was successful

• Less visible progress information

One of the users said that it was great to have all the progress information that appeared as the tool was connecting and saving the configuration from the nodes. But he would have liked more distinct information when everything was completed.
and successful. The other test user felt that it was a bit too much information and only wanted to see that something was happening, but not necessarily what. A link of some sort he could click to see more detailed information would be enough.

Present Page

- Sort columns ascending and descending
- Simple search engine

This page was actually called the search page on the different buttons. The reason for this was that my personal idea was to have a search function, but did not have time to complete this, so a simple listing of all saved configuration was only presented at this page. Both of the users felt that a search engine would be of use and would be nice to have. One of the users said that it could be good enough at first just being able to sort the list ascending and descending on the different columns.

Load Page

- Move the link to view file
- Pre-filled out fields

Two of the tasks, task three and four, were designed so that the users would have to look inside the configuration files. This could be done through the web interface and the link was found on this load page. They both thought that the link should have been at the top of this page and not at the bottom as it is now. One of the users filled out the form before clicking the link, which is not necessary, clearly a design miss. He also said: "I wonder why I have to fill this out" out loud while filling out the form before clicking the link. Even tried afterwards not to fill out the form and clicked the link just to see if it was possible, successfully.

As said before, the users wanted all the forms to be filled out in advance.

More Information Page

- Edit capability

The configuration files are large text files and both the users said that it would be a tremendous help to be able to change the files through the web page. The alternative is to find the file manually in a Unix shell and open it with a text editor such as Emacs or Vi, but this is very time consuming.
Chapter 6

Discussion

6.1 Method

The method I chose to follow and my way of working was good for the scope of this thesis. The company, department and scope of my project met the requirements of the waterfall principle listed in Section 1.4 on page 2. But in a way, I did not follow the method strictly. Since I sat so close to the people who were supposed to use the tool, there were of course continuous discussions that changed my decisions, even if it was in very small steps at a time. I must say that if the project would have been a little bit bigger, or if I would not have sat so close to the team, it felt like an iterative process would have been better.

6.2 Pros and Cons of the Commercial Applications

The applications which exist on the market are very well designed with many useful built-in functions. The user interface is in the cases of commercial applications most often artistic from a graphical point of view. They are relatively easy to use with simple buttons and have drag-and-drop features. But the programs can have too many features and thereby get too complex for the task at hand. The functions, which are useful specifically for the scope of this assignment, include automatic network discovery, configuration of hardware and provide the possibility to get layer 2 and layer 3 information. In some cases, even a graphical layout of the network was a part of the applications, which could be useful.

IBM Tivoli Network Manager Software and HP OpenView Network Node Manager have very good functions for displaying topology mapping. In addition to this, OpenView also support configuration possibilities for many hardware brands. There is also the possibility to buy add-ons to Tivoli, which might make it able to configure hardware through the tool.

Tivoli and OpenView are both built upon the SNMP protocol and require that the hardware is configured for this. Hardware at PRAN does not support
SNMP for the moment and there are no future plans to install this support for the hardware. This fact makes these applications unwanted.

PacketTrap have the most important of the above mentioned functions. The hardware configuration support is particularly good. It has the possibility to save, upload and change hardware configuration through SSH/Telnet.

The downside with PacketTrap is that the configuration support only is applicable for Cisco devices. PacketTrap was the most promising application which could have been of interest for this thesis, but since Ericsson uses other hardware types than Cisco, PacketTrap is unfortunately not an option.

None of the commercial application had features that were ready to use and was good enough for this assignment. In all of these applications it would have been necessary to implement additional functions and a lot of functions would have had to be removed since the hardware used by Ericsson does not support everything. This is normally not possible for commercial applications since the source code is not available.

6.3 Pros and Cons of the Open Source Applications

The obvious advantage with Open Source applications is that the code is available for viewing and editing. This makes it easier to customize the application for a specific need.

In comparison to the commercial applications, the open source applications are not as well developed graphically. The focus has been on having an application that does what it is supposed to do and less time has been put down on making it pretty. This has been an advantage for these applications, it has made them easier to understand. Less options leads to less possibilities of doing something wrong. In the case of this thesis the complexity of the application does not have to be large in order to work as it should, if the proper functions are implemented.

OpenNMS is an exceptional tool for network management and network monitoring. It has the features commercial applications offer and this to a price commercial applications cannot compete with, namely free.

NetDisco is the light version of a normal network monitoring tool. Its strength does not lie in monitoring the network as most applications on the market do, but it focuses on handling switch ports.

Both of these tools are Open Source giving them the advantage of being altered for specific purposes. OpenNMS is the most beneficial of the Open Source application and would also be one of the best choices, considering the tested applications, for small to medium sized networks. Especially if the sole purpose is monitoring the network. Another advantage these tools have is that they are both web-based making it very accessible for users.

These tools do not offer the ability to save the states and configuration files routers and switches have. But since they are Open Source it would be possible to add this functionality to the applications.
6.4 Comparison between Investigated Applications and PRAN Tool

Many of the other features already implemented use the SNMP protocol as base making those functions out of use to Ericsson. Having an application with features that do not work is not good from a usability point of view and it diminishes the credibility of the tool. If for instance OpenNMS would be used as base for the tool wanted by Ericsson, most of the functions would have to be disabled. The functions remaining could be considered a bonus since they are not really requested. But the time and effort that would have to be put down in order to convert OpenNMS to work properly for Ericsson does not make it worth using OpenNMS as base.

6.4 Comparison between Investigated Applications and PRAN Tool

Something that many organizations consider important is the employees’ way of working. Having a tool that does what PRAN Hardware Configuration Tool do can help the employees be more efficient in their daily work or can at least enhance the quality of a product. Their previous way of working requires more manual work and is more time consuming. Using this tool could stimulate more testing and better logging of test cases.

The tool developed for this thesis is tailor made for the basic requirements PRAN have. The applications that exist on the market are more general in some cases, while in some not general enough. Having the possibility to expand to different types of hardware and different types of vendors is more crucial than having all requirements with priority level two ready.

It might be difficult to get a grip of what the applications investigated in this project are capable of. In Table 6.1 on page 48 the different applications, including my own developed tool, are compared with the requirements specified in the beginning of the project. The column named number in the Table 6.1 on page 48 contain a number that corresponds to a description, which is explained in the table below.

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The system is able to generate a configuration document based on the nodes in the network.</td>
</tr>
<tr>
<td>2</td>
<td>The system is able to configure network equipment based on the above mentioned configuration document.</td>
</tr>
<tr>
<td>3</td>
<td>The system is expandable.</td>
</tr>
<tr>
<td>4</td>
<td>The system is graphical.</td>
</tr>
<tr>
<td>5</td>
<td>The system has support for the standard types of nodes that PRAN sells reference solutions for.</td>
</tr>
<tr>
<td>6</td>
<td>The system is able to present uptime and downtime for nodes.</td>
</tr>
<tr>
<td>7</td>
<td>The system is able to save and present firmware information.</td>
</tr>
<tr>
<td>8</td>
<td>The system is able to generate logical topology view based on layer 2 information of the network.</td>
</tr>
<tr>
<td>9</td>
<td>The system support edit functionality in the logical topology view.</td>
</tr>
<tr>
<td>10</td>
<td>The system is able to generate a configuration document based on the logical topology view.</td>
</tr>
<tr>
<td>Number</td>
<td>PacketTrap</td>
</tr>
<tr>
<td>--------</td>
<td>------------</td>
</tr>
<tr>
<td>1</td>
<td>Yes (for Cisco)</td>
</tr>
<tr>
<td>2</td>
<td>Yes (for Cisco)</td>
</tr>
<tr>
<td>3</td>
<td>No</td>
</tr>
<tr>
<td>4</td>
<td>Yes</td>
</tr>
<tr>
<td>5</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>Yes</td>
</tr>
<tr>
<td>7</td>
<td>Yes</td>
</tr>
<tr>
<td>8</td>
<td>Yes</td>
</tr>
<tr>
<td>9</td>
<td>No</td>
</tr>
<tr>
<td>10</td>
<td>No</td>
</tr>
</tbody>
</table>

Table 6.1. Comparison between the different applications

6.5 Programming Languages

The most common programming language that comes to mind when somebody require a platform independent tool is Java. We take Java courses in school, there are tons of documentation and if that is not enough, lots of people (teachers, students, employees at Ericsson) I come in contact with know the language and could be of great assistance in case help is required. A very nice graphical interface is relatively easy to design and the performance, which is often discussed when it comes to Java, should not be a problem for a tool as straight forward as this.

There are many advantages with Java, but all programming languages have drawbacks, otherwise there would be only one language. The purpose of the tool and the programmer’s personal opinion are probably the two main reasons why different programming languages are used in different cases. For this thesis I would like to state a few things that you could call drawbacks. When I say Java is platform independent, it is true, in case Java Sun Software is installed on the computer that runs the application. One thing to think about is also the procedure that has to take place every time the tool has been upgraded. Every computer with an application with an older version has to upgrade the tool every time a new version is available.

I decided to write my program in a scripting language called Perl, and in addition to this, have it done as a web based tool. Having a web based tool written in Perl has of course drawbacks as well. For one, there is the graphical layout which is far from as good as with client-side tools.

One thing that might come to mind when I say that this is supposed to be a web based tool is the aspect of required bandwidth between the server and the client. There should not be any significant difference between a Java solution and a Perl solution. The ‘heavy’ traffic is the traffic between the server and the nodes, which
should be the same for both programming languages. The information passed to the client is only text. The amount of data transferred is quite small and with the data transfer rates in today’s network solutions bandwidth is not an issue.

Furthermore, there are a lot of advantages with Perl. Since the tool is reached through a web page it is platform independent. There is no need for the clients to have any special software other than a web browser since the only instance of the software is on the server. The tool is reachable from any computer, which is connected to the Intranet and no installation is required for each computer. Having the tool only on the server also makes the tool easier to maintain, software updates is only needed to be done in one place. Another advantage from a maintenance point of view is how difficult it is for a new designer to get acquainted with someone else’s code. Even though Perl code can get hard to read, with for instance the regular expressions, it is normally easier to read Perl than Java. This is of course subjective and depends on the knowledge the designer has of each programming language. But Java do tend to get more complex than Perl, for instance with the concept of classes and event handlers.

6.6 Usability Test

6.6.1 Heuristic Evaluation

A heuristic evaluation is normally very useful. It is a simple way to get a lot of common usability problems solved. I did not do a proper heuristic evaluation with a check list and proper evaluators, but in a sense, I did do a simplified version. I motivate this by the fact that I had recently read so much about usability test, many of the common issues where fresh in my mind. I was writing the report while I still was implementing features in the tool making it natural for me to sort the most common problems away. The simplicity of the tool was another reason to why I did not do a proper evaluation. There are not a whole lot of things that can go wrong, making it less useful doing a proper heuristic evaluation. I thought that the think aloud test would catch the problems I missed.

6.6.2 Experiment Test

What I learned from the test was to keep calm while the user was using the tool. Since I designed it I wanted to talk constantly and point out how the features worked. Not because the user was doing something wrong, but because there was a lot of work done behind the scenes which really is not visible. Another thing that made me want to talk was that I knew what to fill out everywhere which gave me the feeling that it took a lot of time for the user. But if you consider that it was the first time he saw the tool it went really fast. It did talk too much so doing this test in advance was a good experience.

The problems brought up after the test can only be considered as problems with low severity. It was not issues that affected the result of the tasks, merely things that could increase user efficiency. The user for this test did comment on the same things as the test users did during the think aloud test. He did not find
as many problems as the others and did not have as many opinions as they did but he did point out the same areas. There could be many reasons for this. That I talked a bit too much might have affected his behavior, even though I talked after he used a feature could give him confidence and thereby affect how he used more features. The fact that he is not an end user nor as into the technical bits as the other testers most certainly affects the result.

6.6.3 Think Aloud

It was very interesting to see that users have so similar reactions to features even though they work in very different ways. One of the test users was very methodical, read the text and made a decision based on facts before acting. The other had a more trial and error approach. Clicking buttons and observing what would happen if he did as he did. It was very nice to see that this usability test works as well as I had read, that this test actually found so many usability problems so easily. Even though the problems were not severe, it is good to get rid of confusing text and irritants in order to increase the positive feeling for the system. I am convinced that the same problems would not have been discovered if I had sat down with each test user and explained all the features prior to the test. The real strength and sincerity towards the usability of the tool is at its best when no previous information about the system has been revealed.

It is quite common that users express more criticism than satisfaction even if a system would be close to perfect. The reason is simple, for most people it is easier to be negative than positive. In Section 5.5 on page 42 I only mentioned the problems with the tool. If I only mention those parts one could get the feeling that there were no positive feelings about the system. This was of course not the case. All test users were very positive to the tool and they all felt great potential in the system. They liked the features and all said that a lot of time can be saved by using this tool, and it was not only this that would be improved. Quality of their product would probably also get improved. A configuration could easily be saved for each test case during their test phase. It would be easier to track problems and thereby correct them. The main feature, automatically saving the configurations was of course very appreciated. They have been saving configuration files a lot before, but never had a good way to store them with filenames, sort them and then find them again. So they found the help the database provides as a very good feature. When it will be possible to search the database it will be even more useful. Loading back the configuration files so easily was very good as well. They found it very easy to skip between test cases when it is so easy to load a new configuration file. When it will be possible to edit the files prior to load through the web interface it will get even more useful.

The usability problems that had to do with poor explanation or confusing text is things I as a developer never really think about since it is so clear to me what the functions do. It is great to have an outside source telling the designer what things are unclear. Other design issues, such as the aspect of password fields that do not hide the characters really did not cross my mind in the beginning. It did not feel very secure to me. But a very good idea was to have a single sign in giving
a user access to the tool. This would prevent wrong users getting access to the
system and thereby making it possible to fill out the login data in advance.

The comments I got about too much progress information is something I think I
will only take lightly into consideration when redesigning the system. All literature
I read say that one should always give the user lots of feedback. But perhaps
redesigning it to feel like less would be an idea. Nielsen also says that all test users
have different opinions, everyone cannot be satisfied at the same time. I have had
too few tests to really know whether it is a common opinion that the tool gives
too much feedback or not. But the general opinion according to the literature is
that feedback is good.

The users succeeded with all the tasks which is very good and one can wonder
why they did that. I do not think that I have designed the ultimate tool with the
best possible design. I would not say that the tasks are too simple in comparison
to what they could be. It was real tasks that end users will perform when the
tool is available for them. I would rather say that the design of the tool was good
enough and that the actions themselves are not complicated. I kept the tool as
simple as possible striving towards the concept that less is more. I was trying
to make a tool that does not need any documentation. I have a design with few
colors, and I keep all dialogues consistent. I minimize the user memory load by
having as much as possible filled out in advance and I give feedback to the user
as the program progresses. Finally I tried to design the tool in such a way that
errors are avoided.
Chapter 7

Conclusions

I am very pleased with my decision I made concerning the first part of the thesis, evaluating whether or not I should develop my own tool or use an existing tool from the market. When all the pieces has fallen into place, when I understood how special this environment is and the task the tool is supposed to do, it is easy to see why a public tool is close to impossible to use. It might have been possible to extend an existing tool with new functionality, but that would have been a lot more time consuming. For starters, getting acquainted and adjusting oneself to someone else’s code can be very hard. Since the PRAN environment is so special, it is also not certain that it would be possible to keep all of the functions from the public tool. But if it would be possible to keep them, they really would not be needed for the primary purpose and would probably confuse the users more than they would contribute. Developing my own tool was absolutely necessary.

Every software developer should make usability tests while they develop their system. It should be a part of their development process. They should at least perform usability tests when they see themselves as finished with the system. I do not think this is the case in today’s companies, at least not in Ericsson. The common thought today is to get something working, and do it fast. But the time saved ignoring usability is lost pretty fast when you sum up the time wasted when people use an ineffective system. A lot of time can be saved and the quality of products can be increased if just a few percent of the projects time is put into usability. Finding so many usability problems, minor and severe, with such easy measures, is not only good to do, but should be mandatory.

It was easy to do usability tests and I really see the benefits of them. If the experimenter has more experience, the tests will be even more useful. The usability test I made exposed some usability problems. Even though the problems where minor, they affect the user performance and should be corrected, especially when these problems are so easy to fix. The focus in this report has of course been on the problems, the purpose of a usability test is to eliminate them. But it is important not to forget the positive feedback I got from the users during and after the tests. They really felt that the tool will be useful.

It still feels like a good decision to have the tool written in Perl as a web based
program. The advantages of having a mobile and easy to maintain program are greater than the advantages you get from a program written in Java. When done with the features and the design, I realize that there is no need for a graphically artistic program. Configuring a web server and writing a web based program that ensure that there will not be timeouts is a bit tricky, but it was worth the effort. The program does what it is supposed to do, and it does it well.

I sometimes refer to this tool as a prototype and you can say that it is. The functions are written for one type of hardware but all the functions are implemented. Adding support for new hardware types will always be an ongoing project since new hardware types always will appear. When that support is added the potential of the tool will increase. To make it even better, some other tool specific features should be implemented, for example the features that were brought up during the usability tests. Some of them were thought of before the test users mentioned them, but due to the scope of this thesis it was not possible to implement them. Hopefully they will be implemented in the future since there is a clear advantage in having them.

As I said, the tool does what it is supposed to do, and it does it well. The remaining requirements, the ones that had priority 2, would be of great assistance if they will be implemented. But the tool is in its present form very useful to PRAN. It will most likely be used during the next test phase according to the staff.
Bibliography


## Appendix A

### Usability Matrix

<table>
<thead>
<tr>
<th>Measurements during testing</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of incorrectly filled out fields or used features (user errors)</td>
<td>[_]</td>
</tr>
<tr>
<td>Number of times user expresses frustration</td>
<td>[_]</td>
</tr>
<tr>
<td>Number of times user expresses satisfaction</td>
<td>[_]</td>
</tr>
<tr>
<td>Number of times user want more progress information.</td>
<td>[_]</td>
</tr>
<tr>
<td>Number of times user request a help page</td>
<td>[_]</td>
</tr>
<tr>
<td>Number of user statements that were positive towards the system</td>
<td>[_]</td>
</tr>
<tr>
<td>Number of user statements that were critical towards the system</td>
<td>[_]</td>
</tr>
<tr>
<td>Number of times the user got stuck at a seemingly unsolvable problem</td>
<td>[_]</td>
</tr>
<tr>
<td>Number of backwards-forward clicks</td>
<td>[_]</td>
</tr>
<tr>
<td>Number of features never used by the user</td>
<td>[_]</td>
</tr>
<tr>
<td>Number of times users couldn’t find their way around the pages</td>
<td>[_]</td>
</tr>
<tr>
<td>Number of suggested design/feature improvement comments</td>
<td>[_]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Measurements after testing</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Users succeeded with 1 of the X tasks</td>
<td>[_]</td>
</tr>
<tr>
<td>Users succeeded with 2 of the X tasks</td>
<td>[_]</td>
</tr>
<tr>
<td>Users succeeded with 3 of the X tasks</td>
<td>[_]</td>
</tr>
<tr>
<td>Users succeeded with Y of the X tasks</td>
<td>[_]</td>
</tr>
<tr>
<td>Number of good features recalled by user</td>
<td>[_]</td>
</tr>
<tr>
<td>Number of bad features recalled by user</td>
<td>[_]</td>
</tr>
<tr>
<td>Number of users preferring the system over other systems or other ways of working</td>
<td>[_]</td>
</tr>
</tbody>
</table>
# Appendix B

## Usability Matrix First Test

### User

<table>
<thead>
<tr>
<th>Measurements during testing</th>
<th>Frequencies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of incorrectly filled out fields or used features (user errors)</td>
<td>11</td>
</tr>
<tr>
<td>Number of times user expresses frustration</td>
<td>1</td>
</tr>
<tr>
<td>Number of times user expresses satisfaction</td>
<td>1</td>
</tr>
<tr>
<td>Number of times user want more progress information.</td>
<td></td>
</tr>
<tr>
<td>Number of times user request a help page</td>
<td></td>
</tr>
<tr>
<td>Number of user statements that were positive towards the system</td>
<td>11</td>
</tr>
<tr>
<td>Number of user statements that were critical towards the system</td>
<td>1</td>
</tr>
<tr>
<td>Number of times the user got stuck at a seemingly unsolvable problem</td>
<td></td>
</tr>
<tr>
<td>Number of backwards-forward clicks</td>
<td></td>
</tr>
<tr>
<td>Number of features never used by the user</td>
<td></td>
</tr>
<tr>
<td>Number of times users couldn’t find their way around the pages</td>
<td>1</td>
</tr>
<tr>
<td>Number of suggested design/feature improvement comments</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Measurements after testing</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>User succeeded with 1 of the 4 tasks</td>
<td></td>
</tr>
<tr>
<td>User succeeded with 2 of the 4 tasks</td>
<td></td>
</tr>
<tr>
<td>User succeeded with 3 of the 4 tasks</td>
<td></td>
</tr>
<tr>
<td>User succeeded with 4 of the 4 tasks</td>
<td>1</td>
</tr>
<tr>
<td>Number of good features recalled by user</td>
<td>1111</td>
</tr>
<tr>
<td>Number of bad features recalled by user</td>
<td>1</td>
</tr>
<tr>
<td>Number of users preferring the system over other systems or other ways of working</td>
<td>1</td>
</tr>
</tbody>
</table>
## Appendix C

### Usability Matrix Second Test User

<table>
<thead>
<tr>
<th>Measurements during testing</th>
<th>Frequencies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of incorrectly filled out fields or used features (user errors)</td>
<td>I I I I</td>
</tr>
<tr>
<td>Number of times user expresses frustration</td>
<td>I I I I</td>
</tr>
<tr>
<td>Number of times user expresses satisfaction</td>
<td>I I I</td>
</tr>
<tr>
<td>Number of times user want more progress information.</td>
<td></td>
</tr>
<tr>
<td>Number of times user request a help page</td>
<td>I I I I</td>
</tr>
<tr>
<td>Number of user statements that were positive towards the system</td>
<td>I I I I</td>
</tr>
<tr>
<td>Number of user statements that were critical towards the system</td>
<td>I I I</td>
</tr>
<tr>
<td>Number of times the user got stuck at a seemingly unsolvable problem</td>
<td></td>
</tr>
<tr>
<td>Number of backwards-forward clicks</td>
<td></td>
</tr>
<tr>
<td>Number of features never used by the user</td>
<td></td>
</tr>
<tr>
<td>Number of times users couldn’t find their way around the pages</td>
<td></td>
</tr>
<tr>
<td>Number of suggested design/feature improvement comments</td>
<td>I I</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Measurements after testing</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>User succeeded with 1 of the 4 tasks</td>
<td></td>
</tr>
<tr>
<td>User succeeded with 2 of the 4 tasks</td>
<td></td>
</tr>
<tr>
<td>User succeeded with 3 of the 4 tasks</td>
<td></td>
</tr>
<tr>
<td>User succeeded with 4 of the 4 tasks</td>
<td>I</td>
</tr>
<tr>
<td>Number of good features recalled by user</td>
<td>I I I I</td>
</tr>
<tr>
<td>Number of bad features recalled by user</td>
<td></td>
</tr>
<tr>
<td>Number of users preferring the system over other systems or other ways of working</td>
<td>I</td>
</tr>
</tbody>
</table>
Appendix D

Tasks for Usability Test

Tasks

1. Save the current configuration from one Redback SmartEdge 800. In the comment field, write something that will make you able to identify your save. The following hardware specific information is needed to complete the task.

<table>
<thead>
<tr>
<th>IP Address</th>
<th>150.132.90.17</th>
</tr>
</thead>
<tbody>
<tr>
<td>Login to router</td>
<td>**********</td>
</tr>
<tr>
<td>Password to router</td>
<td>**********</td>
</tr>
<tr>
<td>Enable password</td>
<td>**********</td>
</tr>
<tr>
<td>IP address to server</td>
<td>172.30.6.31</td>
</tr>
</tbody>
</table>

2. Find your saved configuration file and view the contents of the file. In the file, find the line starting with “system contact”. Note what value that is.

3. Find the saved configuration file that has the comment “Test #1” and find the line starting with “system contact” inside the file. Note what value that is.

4. Go back to the load dialogue and load this configuration (the configuration with comment “Test #1”). Use the hardware specific information from the table above.
Upphovsrätt

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

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

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

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

Copyright

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

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

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

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

© Robert Sandberg