Towards a Development Environment for Model Based Test Design

Han Jing
Abstract

Towards a Development Environment for Model Based Test Design

Han Jing/Sharif Md Rayhan Babu

Within the UP IP I&V organization there is high focus on increasing the ability to predict product quality in a cost efficient way. Test automation has therefore been an important enabler for us. The IP test design environment is continuously evolving and the investigations will show which improvements that is most important to implement in short and long term. In Ericsson UP IP I&V, the test automation framework environments are severed to complete some process by automated method, for improving the efficiency of testing. But there are still many areas or structures existing in the test framework that could be improved. Main purpose of this thesis work is to investigate how the current UP IP I&V test design and execution environment can be improved, in both a short and long time perspective.

In this project, we introduce the basic information that helps beginners to work on with Eclipse environment and Model base test design. In this report we discuss the UP IP I&V test automation environment. We analyze the characteristic of different types of problem and solutions for different exercise. Here we also give some guideline for the system developers and the tester to work properly in this organization.
Acknowledgements

This project work is comprised of lots of contributions by many people. It would not be possible to come to an end without an appreciated help and assistance from them. We would like to convey our deep respect and appreciation to Jan Å Eriksson for giving us the opportunity to work at his unit.

We are grateful to our coach Ann-Marie Eriksson for her enormous help, encouragement and assistance during the project work.

We would like to express our special thanks to Per Erik Westerberg, Ebrahim Amir Khani for his cooperation and assistance during and after this project. We will not forget some of his remarks towards us.

We are deeply grateful to our program coordinator Professor Tomas Lindh, KTH; and Professor Bengt Jonsson, UU; in Sweden and for giving us the opportunity to study in Computer Networks and Computer Science at KTH and Uppsala University.

Special thanks to Jan Arlinde, Gabor Fenyes and Stanislav Vovk.

We would like to thank all the members of UP IP I&V at Ericsson AB for their assistance and cooperation.

Last but not least, we want to acknowledge our family for believing us and encouraging us all the way.

Han Jing

Sharif Md Rayhan Babu
<table>
<thead>
<tr>
<th>Approved</th>
<th>Checked</th>
<th>Date</th>
<th>Rev</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>/EAB/FTP/DNF</td>
<td>Jan Å Eriksson</td>
<td>2008-05-31</td>
<td>D</td>
<td></td>
</tr>
</tbody>
</table>
Table of Contents

Abstract.................................................................................................................................III

Acknowledgements..............................................................................................................V

Table of Contents..............................................................................................................VII

Definitions ..........................................................................................................................1

1 Introduction.......................................................................................................................3
  1.1 General Information....................................................................................................3
  1.2 Background.................................................................................................................3
  1.3 Purpose......................................................................................................................4
  1.4 Prerequisites..............................................................................................................5
  1.5 Related Work..............................................................................................................6
  1.6 Problem Description...............................................................................................6
  1.7 Report Structure.......................................................................................................7

2 Research and Literature Study.......................................................................................9
  2.1 Connectivity Packet Platform (CPP)........................................................................9
    2.1.1 Introduction........................................................................................................9
    2.1.2 System Architecture of CPP.............................................................................9
    2.1.3 IP Transport Service.........................................................................................10
    2.1.4 IP System Model Functions............................................................................10
    2.1.5 CPP Node.........................................................................................................11
      2.1.5.1 GPB..........................................................................................................12
      2.1.5.2 SCB and SXB............................................................................................12
      2.1.5.3 SPB..........................................................................................................13
      2.1.5.4 TUB..........................................................................................................13
      2.1.5.5 ET.............................................................................................................13
      2.1.5.6 CBU..........................................................................................................14
  2.2 Common Expect Test Platform (CETP)...................................................................15
  2.3 Test Instrument.........................................................................................................16
  2.4 Lab Network.............................................................................................................18
  2.5 CETP Test Environment.........................................................................................19
  2.6 Clear Case.................................................................................................................21
  2.7 Unix Shell Script.....................................................................................................22
  2.8 Model Based Test Design.......................................................................................22
  2.9 RSD-RT (Rational Systems Developer-Real Time )...............................................24
  2.10 Conformiq Qtronic...............................................................................................24
2.11 SCTP (Stream Control Transport Protocol) .................................................................26
2.12 Unified Modeling Language (UML).............................................................................27

3 Methodology ..........................................................................................................................29

4 Implementation of improvements in Test Environment.........................................................32
  4.1 Scenario 1- Flexibility of the test environment.............................................................32
      4.1.1 Analysis for improvement of Flexibility test environment.................................34
      4.1.2 Result for Flexibility of the test environment....................................................34
  4.2 Scenario 2- Current structure and distribution of logic in the test environment.............39
      4.2.1 Analysis for improvement of the Current structure and distribution of logic in the test environment.................................39
      4.2.2 Result of the Current structure and distribution of logic in the test environment.......39
      4.2.2.1 Software Description.....................................................................................40
  4.3 Scenario 3 - Tool environment with Model based testing..........................................43
      4.3.1 Analysis for improvement of the Tool environment with Eclipse and Model based test design.................................43
      4.3.2 Result of the Tool environment with Eclipse and Model based testing..................47
      4.3.2.1 Compare to Qtronic Modeler.........................................................................49
      4.3.2.2 Import model Conformiq tool into RSD-RT..................................................51
  4.4 Guideline for system developers & the tester.................................................................59

5 Limitation of the Thesis Work .............................................................................................62

6 Conclusion .............................................................................................................................64
  6.1 Problems .........................................................................................................................64
  6.2 Lessons Learned..............................................................................................................65
  6.3 Future Work.....................................................................................................................65

7 Abbreviations .........................................................................................................................66

8 References .............................................................................................................................69

Appendixes - A ..........................................................................................................................71
Appendixes - B ..........................................................................................................................76
Appendixes - C ..........................................................................................................................88
List of Figures

Figure 2-1 CPP Subrack[4].............................................................................................................12
Figure 2-2 CBU Node[10].............................................................................................................15
Figure 2-3 Common Expect Test Platform (CETP)........................................................................16
Figure 2-4 Agilent N2X................................................................................................................17
Figure 2-5 SmartBits 6000...........................................................................................................17
Figure 2-6 Lab Network[10]..........................................................................................................18
Figure 2-7 I&V Test Environment.................................................................................................19
Figure 2-8 Clear Case View.........................................................................................................21
Figure 2-9. Model Base Testing Procedure ................................................................................23
Figure 2-10. Qtronic Modeler and test generator ......................................................................25
Figure 2-11. SCTP Port Distribution ..........................................................................................26
Figure 5-1. Testing Environment ..............................................................................................33
Figure 5-2. After implementing start variable ............................................................................37
Figure 5-3. Different groups in ipTools .......................................................................................40
Figure 5-4 The Eclipse Platform User Interface ........................................................................45
Figure 5-5. UML Diagrams in Rational modeling tools[23]..........................................................50
Figure 5-6 Qtronic Modeler.........................................................................................................54
Figure 5-7 RSD-RT Moduler in System Boundary ....................................................................55
Figure 5-8 Connection between system and system boundary in RSD-RT.................................56
Figure 5-9 Transition between the state in RSD-RT ..................................................................57
Figure 5-10 Time average in RSD RT and Qtronic ....................................................................58

List of Tables

Table 5-1 Definition Differences Between Qtronic and RSD-RT ................................................52
Table 5-3 Time difference to create a model ..............................................................................57
Definitions

This list of some logic definitions that will help the reader in the following chapters.

Test Object - Test object is a group of function or functionality.

Test Case - A test case is a sequence of discrete test steps. The purpose of a test case is to verify certain functionalities of the test object.

Test Suite - A test suite contains a sequence of test cases that can be executed together.

StartScript - A list of the test cases to run as a test suite.

Verification Specification - A document that defines the functionalities that have to be verified on the test objects for a particular functionality.

CPP Node - A CPP Node is a combination of CPP hardware as well as the software to control that hardware.

CETP - CETP is a combination of the functionalities of Expect/TCL scripting language with the functionalities that are specific to the CPP nodes.

ipTools - ipTools is a TCL API to control the Traffic generators (AgilentN2X and Smartbits) and a library of functions used by the testcases during the verification of the test objects.

ALT – ipTools is an API containing IP functionality and access function to test instruments such as Smartbits and AgilentN2X.

MBT - Model-Based Testing is the automatic generation of efficient test procedures/vectors using models of system requirements and specified functionality.

TPT – TPT stands for Test Performance Tools. To import the test cases result.
<table>
<thead>
<tr>
<th>Prepared (also subject responsible if other)</th>
<th>No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>/EAB/FTP/DNF ESHARAY EHANJIN</td>
<td>/EAB/FTP/DNF- Uen</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Approved</th>
<th>Checked</th>
<th>Date</th>
<th>Rev</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>/EAB/FTP/DNF Jan Å Eriksson</td>
<td></td>
<td>2008-05-31</td>
<td>D</td>
<td></td>
</tr>
</tbody>
</table>
1 Introduction

1.1 General Information

Within the UP IP I&V organization there is high focus on increasing the ability to predict product quality in a cost efficient way. Test Automation has therefore been an important enabler for us. Currently we have about 90% of the system function test cases automated or semi-automated.

We have realized that when new functionality and new standards are implemented in the system, legacy tests can have been impacted due to changed system behaviour in the system under test or in the test framework.

This fact will increase the work load and cause unpredicted rework. It’s of vital importance that we can continue to develop the system without re-working the test legacy. This problem can be addressed through many different improvements both for short and long term.

During 2007 Model Based Test design has been introduced and used as test-design method in one test team in the IP I&V department.

1.2 Background

Ericsson is a world-leading provider of telecommunications equipment and related services to mobile and fixed network operators globally. Over 1,000 networks in 140 countries [1] utilize Ericsson's network equipment and 40 percent of all mobile calls are made through Ericsson's systems. Ericsson is one of the few companies worldwide that can offer end-to-end solutions for all major mobile communication standards. [1]

Ericsson Services Ltd is responsible for the management and operation of fixed line and mobile networks for operators such as '3'. The core of Ericsson's 3G development is the Connectivity Packet Platform (CPP). 3G network components such as Radio Base Station (RBS), Radio Network Controller (RNC) and media gateways are built on the top of CPP.

Testing is an integral part of any research and development process. An efficient testing unit can improve the overall lead time for delivering a product to the end users. The same is true for Ericsson's 3G products. Ericsson performs extensive testing of its 3G technology components.
DU Platforms Product Development Excellence is part of BU Networks Product Development Excellence Program (2007-2010).[2] The program activities contribute to BNET’s overall business objectives as well as Ericsson’s overall objective to strengthen R&D competitiveness by shortening lead-times and increasing efficiency.

DU Platforms and Product Management Platforms, together with all other BU Network’s DUs is represented in the BNET PDE Program Team to ensure a close cooperation between R&D and Product Management on all levels, a key requirement to achieve true PDE success.[3]

The vision for PDU CPP is to be the preferred choice whenever a new or existing application has platform needs. The mission for PDU CPP[2] is to provide a Modeler, flexible and robust platform solution for access and traffic nodes for both existing and new applications.

Inside the PDU CPP organization Ericsson has three more unit CPP Hardware Design, IP Design and User Plan Integration and verification. The UP IP I&V unit of Ericsson (situated in Älvsjö, Stockholm) performs tests on IP functionality in CPP. This unit has a test laboratory where both automated, semi-automated and manual tests are performed on CPP based network nodes. When the new functionality and new standards are implemented in the system, legacy tests often have been impacted due to changed system behavior in the system under test or in the test framework. This fact will increase the work load and cause unpredicted rework. It is of vital importance to continue to develop the system without re-working the test legacy. This is the main reasons why this thesis work was initiated at their premises.

1.3 Purpose

The significance of automated testing is increasing as more and more complex technologies are being developed. Automated testing enables to predict the product quality in a cost efficient way. The same thing is true for all products related to the CPP test at Ericsson. The automation rate is generally very high in all product areas of CPP test which include IP and ATM. But the case with IP part is slightly different. In IP part, some of the test cases have been fully automated, some of the test cases are semi automated and the rest of the test cases have to be handled manually. But due to the fact that IP test are performed for a number of different standards plus the tests make use of a number of traffic generating and monitoring instruments, a lot of manual interventions are needed. It will significantly cut down the total lead time for delivery. It will also allow the utilization of machine hours in a more efficient manner.
In Ericsson UP IP I&V, the test automation environment are severed to complete some process by automated method, for improving the efficiency of testing. But there are still many areas or structures existing in the test framework that could be improved. It will increase the work load and cause unpredicted rework. It is of vital importance that we can continue to develop the system without major re-working of the legacy test. For the sake of brevity a number of test suites have been defined. Each test suite combines a number of test cases. The user can choose to execute a single test case, a number of test cases or decide to execute one of the predefined test suites. Smartbits 6000 or AgilentN2X will be used as a traffic generator, CETP will be used to run automated test cases.

Realizing that when new functionality and new standards are implemented in the system, legacy tests can have been impacted due to changed system behavior in the system under test or in the test framework.

This fact will increase the work load and cause unpredicted rework. It is of vital importance that we can continue to develop the system without major re-working of the legacy test.

In view of the above scenario this thesis work was initiated at EAB/FTP/DNF. The main purpose of this thesis is to develop an environment for current user plane UP IP I&V test design and execution environment can be improved with model based test design and cost for new test case development, in short and long term perspective. There are several improvements that can be done in the short term perspective as well as we expect that we need to get a more integrated test development environment for Model Based Test Design for a long term solution.

1.4 Prerequisites

The target audiences for this report are the managers and testers in the UP IP I&V unit. A general knowledge in telecommunication and scripting language is preferable. Readers who are not familiar with CPP are advised to read chapter 2 of this report before proceeding further.

---

1 Functionality means usefulness or features
2 Legacy means a computer, system, or software that was created for a specific purpose but is now outdated; anything left over from a previous version of the hardware or software.
1.5 Related Work

As far as Ericsson is concerned, no detailed research relating to the topic of this thesis has been performed in their promises. None of the Ericsson Testing Environments have an existing setup that is similar to the one proposed in this thesis. As most of the technologies related to the thesis work are proprietary to Ericsson, there is a very little chance that similar work has been performed elsewhere.

1.6 Problem Description

- **Inflexibility of regression test environment.**

  When new functionality and new standards are implemented in the system, legacy tests can have been impacted due to changed system behavior in the system under test or in the test framework.

  This fact will increase the work load and cause unpredicted rework.

- **Unsystematic code structure in ipTools makes the complicate configuration setting parameter in testing work.**

  Analysis the current structure and distribution of logic in the test FW. This activity aims to analyse the current structure of the test framework regarding the fact that new target HW variants, new test instruments API’s and standards will be introduced and not possible to run on same test suit.

- **Instability situations happen during the regression test and daily test, short of compatible and robustness in the test instruments.**

  The testers mentioned a very similar problems when we asked what the instability problems in the test framework environment. Mostly, when tester run lots of test cases in the night, some test case passed, some test case failed, it’s very normal situation. But a simply problem happen frequently, for example, when tester run 100 hundred test cases for fifty times, some test cases( we can say about like 10 test cases) always failed in the first twenty times or thirty times, but pass in the next ten times and twenty times. Then many tester consider these problems as instability in the test framework environment.

- **Maintainability of ipTools regarding the current structure and distribution logic.**

  As time goes by, the code and structure, as well as the distribution of logic in this file becomes too large and complicated to maintain and update.
The SCTP packet procedures in ipTools don’t work for both AgilnetN2X and Smartbits.

Smartbits and AgilentN2X are two main test instruments in test framework, when some test cases, network protocol testing (for instance, SCTP packet procedures in ipTools) or/and some test functionality can only run on Smartbits or AgilentN2X due to the limitation of compatible and robustness. Its aspects will be needed more Fault Tolerant.

Integrated tool environment with RSD-RT and Qtronic test generator.

During 2007 Model Based Test design has been introduced and used as test-design method in one test team in the IP I&V department. In the future, the Model Based Test will be the main testing method in the organization. And the main tools will be RSD-RT and Qtronic test generator.

1.7 Report Structure

The rest of the report is structured in the following way

Chapter 2 describes related background information on CPP and technologies and standards relevant to the thesis work.

Chapter 3 describes the proposed improvements found to reach the thesis target and also describe our thesis goals in a more detailed manner.

Chapter 4 describe the problem analysis result and solution implementation according to the proposed solution in chapter 3 and some guideline for the system developers and the testers.

Chapter 5 analyzes some limitation of the thesis work

Chapter 6 describe the conclusion of the thesis work.
2 Research and Literature Study

This section provides the general information on various standards and technologies that must be understood in order to suggest improvements. For the external version, this part is not necessary for reading at all details; if you are not interested in some technologies, please feel free to skip some parts of this sections since it will not affect you understanding our research and work.

2.1 Connectivity Packet Platform (CPP)

2.1.1 Introduction

Connectivity Packet Platform (CPP) is a scalable platform payload processing node.[5] CPP enables a high degree of redundancy and robustness in order to suit high-availability applications. Based CPP it is possible to develop a packet handling network node, e.g. an ATM Switch, a Radio Base Station, a Radio Network Controller or a Media Gateway.[4]

CPP is one main project in Business Unit Networks (BNET) which focuses on developing leadership in next generation converging networks, by leveraging our leading positions in mobile and fixed access, transport and core. BNET is a global organization, with more than 21 000 employees. CPP is widely used in Ericsson to provide solution on Network Infrastructure for any communication need over mobile or fixed connections.

2.1.2 System Architecture of CPP

In CPP system architecture signaling services offer applications the means to establish and release connection in an Time Division Multiplexer (TDM), Asynchronous Transfer Mode (ATM) or an IP transport network and to exchange signaling messages between applications within a signaling network.

---

2.1.3 IP Transport Service

IP Transport Services enable communication with applications, using IP for transport, that are part of CPP nodes as well as communication through IP with External equipment that has been attached to CPP nodes through external interfaces. [4]

The CPP node component provides IP transport for three different areas

IP Based Operation & Maintenance (O&M)
IP Based SS7 Signaling
IP Based Payload

2.1.4 IP System Model Functions

In UP IP I&V there are various IP system model function as follows :

IPBSA

The IP Bearer Services Access (IPBSA) system function provides Internet Protocol (IP) based transport services to user applications.[9] This includes:

• IP Hosts that applications can use for data transport.
• Interface for control and monitoring of IP Hosts.
• Interfaces and functions for operation and management of IP Hosts.
• Raw IP socket interface.
• A service for user data transport via UDP (User Datagram Protocol)\(^5\) based sessions. Via control plane interfaces these sessions can be set up, released, modified and monitored. [9] Via data plane interfaces applications can send and receive data. Optionally, UDP sessions can be enhanced with RTP (Real Time Protocol) information and RTCP (Real Time Control Protocol) supervision.

• ICMP (Internet Control Message Protocol)\(^6\) functionality. [9]
• Support for network synchronization using the SNTP (Simple Network Time Protocol). [9]

IPG

---

\(^5\) User Datagram Protocol is one of core protocol of internet protocol suites, http://www.faqs.org/rfcs/rfc768.html
\(^6\) Internet Control Message Protocol to send error message, http://www.faqs.org/rfcs/rfc792.html
The IP over Gigabit Ethernet (IPG) system function handles Layer 1 and Layer 2, i.e. Physical and Data Link layer\(^7\), for IP version 4 (IPv4)\(^8\) transport over a Gigabit Ethernet (GE). The GE interface is VLAN\(^9\)-aware, i.e. it is capable of recognizing and generating VLAN-tagged Ethernet frames.

**IPD**

The primary purpose of the IP Distribution system function is to provide distribution of IP version 4 (IPv4) traffic between hosts in the IP Bearer Service Access system function (IPBSA) and an IP network.

IP packets can be distributed to IP hosts defined in the IPBSA system function. [14] A user registers with IPD and is then associated with a distribution handler located at intended distribution point. The traffic interface of IPD toward the link layer is either to the IPG system function (for Gigabit Ethernet link) or to the IPF system function (for Fast Ethernet link).

The IP hosts can be located on General Processor Boards (GPBs) or Special purpose Processor Boards (SPBs) or ET-IP boards. [14] IP distribution to IP hosts on ET-IP boards can only be done to IP hosts located on the same board as the IP distribution function is located.

When an IP host is registered with IPD to a distribution handler, its IP address is put into IPD's distribution table for that handler.

### 2.1.5 CPP Node

A CPP node consists of a number of subracks and each subrack contains a variable number of boards. A board is either a General purpose Processor Board (GPB) or a device board such as Switch Core board (SCB), Switch Extension board (SXB), Timing Unit Board (TUB), Special purpose Processor board (SPB), ET board (ETB) and Application Device board (ADB). All boards are attached to the backplane. [4] A standard CPP subrack consists of 28 slots.

---


\(^{9}\) VLAN stands for Virtual Local Area Network, http://www.faqs.org/rfcs/rfc3069.html
A single subrack node contains one subrack. The subrack acts as the main subrack. A multiple subrack node consists of up to 9 subracks. One subrack acts as the main subrack and the other subracks act as extension subracks.

### 2.1.5.1 GPB

Since a Generic Processor Board (GPB) already include processing and storage capacity it only needs to be complemented with a switch interface and a LED\(^{10}\) display. [4] The purpose of the GPB is to add processing capacity to the main processor cluster and provide persistent storage. The GBP offers the following:

- Ethernet (10/100 BaseT) and serial interfaces (RS232) [4]
- HW for AAL5\(^{11}\) segmentation and reassembling
- HW for AAL1 and MTP2[4]
- Disk HW, Flash disk size up to 1.5 GB[4]

### 2.1.5.2 SCB and SXB

The purpose of a Switch Core Board (SCB) and Switch extension Board (SXB) is to provide a reliable inter board/subrack communication. The SCB/SXB contains the following.

- Inter-subrack link interfaces[4]

---

10 LED stands for Light emitting diode is a semiconductor diode
11 AAL5 is ATM Adaptation layer 5, http://www.faqs.org/rfcs/rfc2364.html
System synchronization distribution

Power feeding filter (-48 Vdc (direct current) for the subrack, compatible with both high and low ohmic distribution. (SCB only) [4]

Serial Bus (RS485) for connecting to external equipment (SCB only) [4]

2.1.5.3 SPB

The purpose of Special Purpose Processor Board (SPB) is to treat payload data that is not suitable for processing in a DSP. A SPB contains the following:

3 SPMs (SPB) or 5 SPM (SPB2) [4]

Each SPM contains a PPC 750-based processor[4]

HW support for ATM/AAL user data termination.

2.1.5.4 TUB

The purpose of a Timing Unit Board (TUB) is to provide a stable system clock pulse to all boards in the system and provide a reference signal for base station application (distributed in special lines in the back plane. A TUB contains the following:

TU common part for the system clock (19.44 MHz) generation[4]

Application specific part for RNC and RBS specific functions

Input for the dedicated 1,544 MHz, 2,048 MHz, and 10 MHz timing reference[4]

2.1.5.5 ET

The functionality of an ET boards is split into three parts: DBM, Framing and multiplexer/demultiplexing HW and a physical port-specific part. [4]

- TDM Exchange Terminal

The TDM exchange terminal is used to connect to TDM networks. Device hardware converts individual channels\(^{12}\) into internal AAL1 streams.

\(^{12}\) Channel works for signal amplification paths.
• ATM Exchange Terminal

The ATM exchange terminals are used to connect to ATM networks or to transfer ATM traffic through a TDM network. Specific HW is used to handle ATM cells and multiplex\textsuperscript{13} and de-multiplex AAL2 traffic.

• Ethernet Exchange Terminal

The Ethernet exchange terminal is used to connect to an IP network using Gigabit Ethernet.

The ET-MFG board has protection switching against link failures, thus only one of the link ports are active at the time. \[4\]

The ET-MFX11 (Intended for RBS) provides six 10/100/1000Base-T\textsuperscript{14} ports and one 1000Base-X\textsuperscript{15} for SFP use. \[4\]

The ET-MFX12 (Intended for RXI and RNC) provides six 10/100/1000Base-T ports and one 1000Base-X for SFP use. \[4\]

ET-MFX13 (Intended for RXI and RNC) provides six SFP ports and one single electrical port. 10/100 Mbit Ethernet connections is available on CBU & GPB. \[4\]

2.1.5.6 CBU

The Control Base Unit board is a multi-function board and contains functionality from GPB, SCB, TUB and ET-MC1. This board is used in a smaller subrack type containing 24 slots and is used for RBS applications. \[16\]

\textsuperscript{13} Multiplexer is a device for combining several channels to be carried by one line or fiber.


\textsuperscript{14} Base T is a Ethernet cabling type to introduced increase reliability and allow the use of existing twisted-pair cable.

\textsuperscript{15} Base X Ethernet cabling type for gigabyte Ethernet
2.2 Common Expect Test Platform (CETP)

Common Expect Test Platform (CETP) combines the functionalities of the Expect/TCL script language with the functionalities that are specific to the CPP nodes. This combination provides a framework for the test engineers for developing CPP specific automated test scripts. CETP is implemented as a shell.

The test script implementation using CETP allows automated test cases to communicate with one or several cello nodes and with external instruments. There is no need for compiling the test scripts before running the test cases since Expect is an interpreted scripting language.

Through CETP, users can write automated tests, especially those tests that are repetitive in nature. All tests are executed from UNIX terminal and when the test is finished, the user gets a summary of the commands that have been sent and received. The user also gets a report which lists the number of tests that have been passed or failed. At the same time three log files in three different formats are created with all events logged - one in a text format, one in HTML format and the other one in XML format (refer to Figure 2-3).

---

16 Except/TCL is a scripting language
17 XML is a markup language much like as HTML
2.3 Test Instrument

In UP IP I&V has two main test instrument in the lab:

**AgilentN2X**      **Smartbits**

These two test instrument of UP IP I&V are mainly used:

- to generate and analyze IP traffic such as UDP, RTP\(^{18}\) and RTCP\(^{19}\)
- to run conformance tests on standard protocols
- to measure the throughput

In general AgilentN2X is used for voice, video and data (triple-play) of next-generation network devices.[17] But in UP IP I&V, AgilentN2X is used to simulate and analyze SCTP\(^{20}\), RTP, and RTCP traffic.[18] This instrument also allows for the generation of Code(interface) that can be used by ipTools. AgilentN2X has an advantage when creating procedures and scripts compared to Smartbits (comparison about the advantage will be presented later in this section). Figure 2-4 shows the AgilentN2X, which is used in UP IP I&V lab.

---

\(^{18}\) RTP is a transport protocol for real time applications, http://www.ietf.org/rfc/rfc3550.txt

\(^{19}\) RTCP is real time transport control protocol, http://www.ietf.org/rfc/rfc1889.txt

\(^{20}\) SCTP is a transport layer protocol, http://www.ietf.org/rfc/rfc2960.txt
In general, Smartbit is used for the high port solidity testing of Gigabit and 10Gigabit Ethernet, ATM, Frame Relay networks and network tools examination. [19] Smartbits is replaced by Spirent Test Center and will not be further developed by the tool provider. UP IP I&V has not decided whether they will invest in Spirent Test Center. Figure 2-5 shows the Smartbits 6000 being used in the UP IP I&V lab.

Both Smartbits 6000 and AgilentN2X are used as traffic generators. The N2X traffic generators will be available in both 1000Base and 10GBase versions. The 10GBase is only used for characteristic tests. Compared to AgilentN2X, Smartbits has a faster setup test configuration than AgilentN2X, also the API is easily adapted to ipTools, and Smartbits documentation is better than AgilentN2X. AgilentN2X has been chosen as more future tool as new features are developed for it because of it’s more powerful and more advanced than Smartbits, see below info.

From the UP IP I&V point of view

- When using the test setup for testing RTP/RTC only AgilentN2X can be used.
- When use the test setup for testing more than 4500 sessions or other characteristics tests only AgilentN2X can be used.
In test setup, for all normal testing that does not include any RTP/RTCP and when AgilentN2X cannot be used, then only Smartbits can be used.

Spirant Test Center looks very promising from a specification point of view but has a totally different API\(^2\) structure which will have great impact on ipTools etc if Ericsson UP IP I&V decide to use it.

### 2.4 Lab Network

In the UP IP I&V lab system are connected with two test instruments, Smartbits and AgilentN2X as shown in Figure 2-6.

---

**Figure 2-6 Lab Network**

---

\(^2\) API is a source code interface that an operating system, library or service provides to support requests made by computer programs.
Figure 2-6. Shows how the workstations, nodes and test instruments are connected to each other. The section of 'LAB' in the above picture is called Internal Connection in laboratory environment. Sniffer\(^{22}\) port is used for a PC as a sniffer, located in the Switch. Smartbits 6000 or Agilent N2X will be used as a traffic generator. PC(N2X-GUI) \([10]\) is a GUI (Graphical user interface) program in a PC to control AgilentN2X.

The section of 'CONTROL ROOM' is Internal Connection of Ericsson's Computer and WS is a Workstation, PC(SB-GUI) is a GUI (Graphical user interface) program in a PC to control Smartbits. DNS is stands for Domain Name System, DNS Server version are BIND\(^{23}\) 9.3.1\([10]\) and NTP is stand for Network Time Protocol.

In this test environment, T-Ethereal\(^{24}\) (preferably) will be used for packet analyzing. Here notice that Common for all Test Setup with Function Test node is that RS232 interfaces on the GPB boards in slot 10 and 11 (not shown in the figures) must be connected to the Terminal Server. \([13]\)

### 2.5 CETP Test Environment

At UP IP I&V, the CETP environment is as follows

---

\(^{22}\) Sniffer is a program that monitors and analyzes the network traffic  
\(^{23}\) BIND is the most commonly used DNS server on the Internet  
\(^{24}\) T-Etheral is a network protocol analyzer
Figure 2-7 shows how the test environment works for function test and regression test. Here we know that the CETP is the combination of TCL & Expect and connected to its configuration file and start script. CETP config file is connected to testing software’s ipTools and other library and test cases. Start scripts consist of a list of test cases.

**ipTools**

ipTools is a TCL API to control the cello\textsuperscript{25} packets platforms, Traffic generator, AgilentN2X & Smartbits.

This file (ipTools.exp) contains the functions implementing IP functionality and access to test instruments such as Spirent's SmartBits and AgilentN2X's. ipTools is also a library including IP related procedures or functions.

**CETP.cfg**

CETP.cfg is a configuration file which determines the behavior of CETP. This file defines the start values for variables (~constants) to be used when running a test case or a test suite. Examples of these are:

- host name for the node to use
- IP address and port number for the test instrument to use
- paths where scripts and procedures are to be found
- slot positions for different board types
- location where the results should be saved

**Startscript**

This file contains a list of the test cases to run as a test suite.

---

\textsuperscript{25} CELLO is Ericsson connectivity packet platform
2.6 Clear Case

Clear Case is a Software Configuration Management (SCM) system. That means it keeps track of which versions of which files were used for each release (even internal releases) and also which combinations were used in builds (including engineer builds, nightly and release builds).[27] Clear Case also provides a rich and robust set of tools to allow a company to craft their engineering environment to their own requirements.

In UP IP I&V, testers have the TIV (Transport Integration Verification) VOB\(^\text{26}\) in Clear Case. The test environment is controlled by the test software which is a collection of scripts that are stored in appropriate locations in the VOB. It means users store test scripts and config files etc in the TIV VOB, and also there locate test software and applications.

![Clear Case View](http://searchsqlserver.techtarget.com/sDefinition/0,,sid87_gci959448,00.html)

---

\(^{26}\) VOB is centralized database that stores version information about the file and the folders in a software configuration management system. The term is usually associated with ClearCase.

http://searchsqlserver.techtarget.com/sDefinition/0,,sid87_gci959448,00.html
2.7 Unix Shell Script

Shell Script is a command line interpreter of an operating system. [26] In general, it is a scripting language written for the Unix shell. Moreover, it is considered as a domain-specific programming language. File manipulation, [26] program execution and printing text are the most frequent operations performed by shell script. In addition, it is much easier and quicker to write a shell script than other equivalent programming language or scripting language. Command.com which is used for DOS and cmd.exe for windows are also command line scripts but they are known as batch files.

In our thesis work we used a shell script when implementing one of the improvements. We chose shell script since they do not need to be compiled, so the script can be executed quickly while debugging.

2.8 Model Based Test Design

Model Based Testing is a software testing[27] method. [21] When using model based testing test cases are derived in whole or in part from a model that describes the functionality or requirements of the system to be tested. In UP IP I&V model based testing is describe in the following way:

Model – formalizes the behaviors of the system, e.g. state machine

Based – generate test cases automatically from the model

Testing – offline or online, need interface between the Test Environment (TE) and the generated output

---

[27] Software Testing is the process of executing a program or system with the intent of finding errors. 
http://www.ece.cmu.edu/~koopman/des_s99/sw_testing/
Figure 2-9. Model Base Testing Procedure

Model Based Testing can greatly help in these areas:

- no need to write separate verification specification (VS\textsuperscript{28}),
- easy to generate new TCs when legacy is affected,
- easy to change the model when a bug in the requirements, specifications or model is found, a new regeneration is easily done,
- easy to mask not implemented functions in the model or library,
- savings and reuse in the long run when there is a good MBT-base,
- shorter schedules, lower cost, and better quality.

\textsuperscript{28} VS is to show that a product/product model (in each phase of the development) will fulfil requirements specified in projects product specification.
2.9 **RSD-RT (Rational Systems Developer-Real Time)**

Rational Systems Developer is the name of a collection of Test and Development Tools invented and published by IBM. [23] It is designed to support model driven development, which is the development of the appropriate models to facilitate all development activities and stages in the system development lifecycle, including tools to transform models into more detailed or more abstract models to move development forward.

In the system architectural analysis area, Rational Systems Developer is very good at system architectural analyzing and simulation, users can delineates the logical subsystems of the system and the initial deployment or locality model in UML 29, which means users can use UML diagramming[23] to elaborate on the architecture in a model driven development environment.

For this feature in RSD, we can also get code with the assistance of visual development tools transforming our model or set of models to platform-dependent implementation models.

2.10 **Conformiq Qtronic**

Conformiq Qtronic is a design model driven test automation tool for automatic test derivation, execution and analysis. [22] When users give a “design models” of the system as an input, Conformiq Qtronic can generate test scripts for the system automatically.

Conformiq Qtronic can automatically generate and execute tests based on design models.

Technically speaking, we know online testing(Online testing means that a model-based testing tool connects “directly” to a system under test and tests it dynamically.) and offline script generation(Offline generation of executable tests means that a model-based testing tool generates test cases as a computer-readable asset that can be later deployed automatically.) are two complementary methods that have both their own strong points. But it is clear that in many traditional software processes, external test scripts bring in benefits on the process management level and Conformiq Qtronic meets this need.

---

29 UML is a standardized visual specification language for object modeling
Figure 2-10. Qtronic Modeler and test generator

So, Conformiq Qtronic supports both online testing\(^{30}\) and offline test\(^{31}\) case generation, [31] both based on models and this module has some benefits are as follows:

- TC and identity could be changed between different TC-generations
- Support for incremental generation of TCs to avoid regeneration from the whole model
- Easier control and better feedback when the tool is generating TCs. This phase can be time consuming and give the feel of little progress

---

30 Online Testing means that a model-based testing tool connects “directly” to a system under test and tests it dynamically.
31 Offline Test case generation of manually deployable tests
2.11 SCTP (Stream Control Transport Protocol)

SCTP is an Internet Protocol (IP) transport protocol. A function that provides SCTP transport for user application on a processor is denoted an SCTP protocol entity. An SCTP protocol entity is tied to an IP host function that provides the IP layer transport\[11\].

In the present implementation of SCTP, an SCTP protocol entity is always tied to an IP host located on a General Purpose Processor Board (GPB). [11] This means that on every Main Processor (MP) where there is an SCTP protocol entity. Since each IP host function has its own IP address, the number of IP addresses needed for a CPP node will increase when more MPs are added to increase the capacity of the node. For nodes that have ET IP hosts, these can not be used for SCTP transport. Separate host functions are needed on the MPs.

![Figure 2-11. SCTP Port Distribution](image)

32 IP Layer Transport is the first three layer of the OSI model
The proposed solution is based on that the ET IP host is used for SCTP transport but the SCTP protocol function is still located on GPBs. SCTP protocol functions on multiple GPBs can use the same ET IP host.[11] It is also possible for SCTP protocol entity to use two alternative ET IP hosts for redundancy.[33]

2.12 Unified Modeling Language (UML)

The Unified Modeling Language (UML) is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing object-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects. Using the UML helps project teams communicate, explore potential designs, and validate the architectural design of the software.

Goals of UML:

Provide users with a ready-to-use, expressive visual modeling language so they can develop and exchange meaningful models.

Provide extensibility and specialization mechanisms to extend the core concepts.

Be independent of particular programming languages and development processes.

Provide a formal basis for understanding the modeling language.

Encourage the growth of the OO tools market.

Support higher-level development concepts such as collaborations, frameworks, patterns and components.

Integrate best practices.

[33] Redundancy is the quality of a system, an item of information, or a bit that is redundant. 
<table>
<thead>
<tr>
<th>Prepared (also subject responsible if other)</th>
<th>Checked</th>
<th>Date</th>
<th>Rev</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>/EAB/FTP/DNF ESHARAY EHANJIN</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>/EAB/FTP/DNF Jan Å Eriksson</td>
<td></td>
<td>2008-05-31</td>
<td>D</td>
<td></td>
</tr>
</tbody>
</table>
3 Methodology

A Test Automation Framework is a set of assumptions, concepts and practices that provide support for automated software testing. In Ericsson UP IP I&V, the test automation environment are served to complete some process by automated, for improving the efficiency of testing. But there are still many areas or structures existing in the test framework that could be improved which increase the work load and cause unpredicted rework.

It is of vital importance that we can continue to develop the system which means the project in CPP IP I&V, like daily testing, function testing requirements without major re-working of the legacy test. For the sake of conveniency a number of test suites have been defined. Each test suite combines a number of test cases. The user can choose to execute a single test case, a number of test cases or decide to execute one of the predefined test suites. Smartbits 6000 or AgilentN2X will be used as traffic generator, CETP will be used to run automated test cases.

In UP IP I&V, some of the test cases have been fully automated, some of the test cases are semi automated and the rest of the test cases have to be handled manually. But due to the fact that IP test are performed for a number of different standards plus the fact that the tests make use of a number of traffic generating and monitoring instruments, a lot of manual interventions are needed. For instance, like regression testing need some configuration setting by manually changed the variables or arguments in some test cases and test suits before implementation. Automation will also allow the utilization of machine hours in a more efficient manner.

In view of the above scenario and of the problem given at section 1.6. After discussing the possible improvement with the management of UP IP I&V decided to improve the following three areas:

- **Inflexibility of regression test environment.**

Because when new functionality and new standards are implemented in the system, legacy tests (regression test) can have been impacted due to changed system behavior in the system under test or in the test framework. That’s the main technical difficulty why the regression test environment is so inflexibility. In Ericsson, it means when new hardware or software embedded into our regression test environment, then it will cause a lot of reworking and cost.
And regression test is the main testing work in UP IP I&V, it takes almost 50% human time and machine time to run legacy everyday which is also called ‘daily test’. It means if we can improve the flexibility of regression test environment, we can decrease most percents of work time spending on daily testing work

- **Maintainability of ipTools regarding the current structure and distribution logic.**

Because ipTools is the main test tool in the regression test and function test used by tester everyday, which means it has hundreds of procedures and functionalities. And the current structure and distribution logic is not so good for one person maintaining, and it costs a lot of time to rewrite some procedures when the tester doing this based on so inorganized structure code. If we can improve the maintainability of ipTools, then tester will be able to easy change and build new procedures or functionalities much quicker than before.

- **Integrated tool environment with RSD-RT and Qtronic test generator.**

Because during 2007 Model Based Test design has been introduced and used as test-design method in one test team in the IP I&V department. For the future testing work, if we can integrate tool environment with RSD-RT and Qtronic test generator together, it can reduce a lot of cash flow and time spending on the testing work.

In order to implement these three improvements and to reach the thesis goal we need to take the following steps:

- Investigate the functionalities and the interfaces used by the instruments and the CPP (Connectivity Packet Platform)
- Get acquainted with methods of automated testing using Expect/TCL on the CPP Expect Test Platform (CETP)
- Analyze strategic and logical grouping of automated test scripts and define longer suitable test suites that can be run over night or over the weekends without manual interaction
- Define methods and procedure to design test scripts that will be used in ipTools environment
- Implement scripting program for change of the execution APIs
- Implement the new procedures to allow start variables to be over-ridden.
• Restructure the test software regarding the current structure and 
distribution of logic in the test environment

• Configure the Eclipse environment for current TCL design with 
ipTools and CETP in different OS\textsuperscript{34} platform

• Implement the RSD-RT model in Eclipse environment and 
introduce a UML modeled in the Qtronic compiler

• Summarize and evaluate the results and suggest further 
 improvement possibilities to the UP IP I&V test environment

• Suggest methods for handling the usage of limited resources in UP 
IP I&V test environment

\textsuperscript{34} OS means Operating System. An operating system is the program that, after being initially loaded into the 
computer by a boot program, manages all the other programs in a computer
4 Implementation of improvements in Test Environment

This section describes the details of implementing the improvements identified. The implementation phase followed the recommendations set forward in the methodology describe in section 3. The purpose of building the test environment is to provide a flexible and maintainable and reducing the cost for new test case development in both a short and long time perspective.

4.1 Scenario 1- Flexibility of the test environment

A typical test case in IP test is performed on different exchange terminal boards and on several different types of CPP nodes using traffic generating and monitoring instruments. The end result is that each test case is executed for many of different combinations (like needing to run in different test instruments or different values).

The start variable values (such as board type, test instrument) are defined in CETP.cfg and this file cannot be changed during the execution of a regression test and function test suite. During the test execution tester cannot change the value because in CETP config file it is impossible to change any value and there are approx thousand of test cases so without changing CETP config file we have to improve the test environment.

Therefore, when composing test suites consideration needs to be taken regarding the test instruments and board types to be used. The result is a large number of test suites making maintenance of these more difficult.

The suggested solution is to introduce intelligence into startscript.exp which will allow the start values to be over-ridden.

35 Regression test is the process of testing changes to computer programs to make sure that the older programming still works with the new changes. http://searchsoftwarequality.techtarget.com/sDefinition/0,,sid92_gei212884.00.html
In the above Figure 5-1 shows a test suite containing two test cases. When TC1 (Test case 1) running, in the code of TC1, there are many variables. And for instance, one value in the TC1 means a IP Address like showing in the picture, in order to run the test case, it must request for a value of Base IP Address in CETP.cfg (CETP config file), after sending the request to CETP, the CETP.cfg file will check the value and return value 192.168.0.1 as the Base IP Address, now you can see it’s because this value is set in CETP config file.

But in CETP config file, it can not compare the value with others because there is no global variable value set yet and in CETP config file we can not change any thing, we can not add value in manually. So CETP config file returns the value of Base IP Address into TC1.

After TC1 getting the value of Base IP Address, in the picture, you can find in the test suites, now goes through another procedure.

Finishing operation with TC1, test suites will automatically run TC18 (Test case 18) and this test case also needs a value, so it requests for a value of Base IP Address in CETP.cfg.
This request sent to CETP and CETP.cfg check the value and return value 192.168.0.1 as Base IP Address, because 192.168.0.1 is the default value which is written in CETP.cfg config file.

According these processing explanation, during the test suite execution you see we can not change the value of Base IP Address.

The proposed improvement will allow the testers to compose regression test suites without having to take consideration to board types or test instruments needed. This will result in more efficient use of the nodes since the suites can be designed to run during the night or at weekends.

4.1.1 Analysis for improvement of Flexibility test environment

Regarding the flexibility of the test automation environment we come up the following improvement need to reach the goal.

- The scope for the flexibility improvement of the automation framework environment will be scripts.
- Three new procedures will be introduced to the ipTools file which will implement new functions in order to make test framework more flexible.
- Test cases must be modified to use ipTools instead of CETP
- Shell script created to facilitate this modification
- Set the start variable to change the default value

4.1.2 Result for Flexibility of the test environment

Making intelligent in test suites after setting the start value ipTools check first is there any valuable value is globally set or not, if the value is set then return the value from ipTools, if not then its take value from the CETP config file. According to doing this, we must apply some new functions and procedures in the ipTools.exp file. These new procedures will use CETP script to run function test and regression test and the configuration of the test environment CETP script does not run in all regression test, however for these regression test use CETP script mostly use the same node configuration as function test.

- Implementation of New Procedures in ipTools

---

[IEEE 90] Flexibility is the ease with which a system or component can be modified for use in applications or environments other than those for which it was specifically designed [IEEE 90].
If all the test suites can be integrated in some way or the startscript.exp file can be intelligent then the test cases can be executed sequentially even they are needed to run on different instruments. Depending on the user input the appropriate test script begins to execute. The script gathers the necessary test cases and /or test suites from the VOB that were selected by the user. It will allow the utilization of machine hours in a more efficient and flexibility manner.

- **Introduce three new procedure**

Three new procedures will be introduced to the ipTools file as follows:

**getCfg**

PROCEDURE: ::ipTools::getCfg

Parameter:

aCfgName  - Name of config variable.

Returns a configuration value previously read from the config file otherwise it returns a configuration value previously set from the global variable.

**setCfg**

PROCEDURE: ::ipTools::setCfg

Parameter:

aCfgName  - Name of the Variable.

aCfgValue - Value to set.

It sets the value that are globally exists.

**unset**

PROCEDURE: ::ipTools::unSetCfg

Parameter:

aCfgName - Name of the Variable.

It unset the value that are set before.

With implementing three procedures, we need to make startscript ‘intelligent’.
Firstly we have to modify all of the test cases to use `::ipTools::getCfg` instead of `::CETP::Cfg`. That’s why we need to create a shell script to facilitate this modification.

This shell script will change all of the test cases that users have to execute.

Before the improvement, variables in the test cases take values from the CETP config file but it’s not possible to change value during the test suite execution.

But in ipTools it’s possible to set the variable value during the test suite execution, this is the reason that we modified all of the test cases to use `::ipTools::getCfg` instead of `::CETP::Cfg`.

Now we have to set and unset the variable value in the startscript.exp. In our three new procedures the `::ipTools::getCfg` works for the test software ipTools and the `::ipTools::setCfg` and `::ipTools::unSetCfg` procedures work for the startscript.

After setting the variable value and unsetting the variable value tester can run the test cases in the test suites.

With implementation of the new procedures in ipTools.exp file, the new distribution of ipTools, CETP, Startscript and Test case are shown in Figure 5-2
Figure 4-2. After implementing start variable

After setting the start variable it follows the different steps

Step 1 – Test Suites starts executing from the Test suites in TC1 and TC1 needs a variable value

Step 2 – In the TC1, it request a value as Base IP Address which needs to get from ipTools

Step 3 - Since Base IP Address does not exist as global Variable value so ipTools fetches the value from CETP.cfg for TC1.
Step 4 - Then CETP.cfg check the variable value and return Base IP Address = 192.168.0.1

Step 5 - Now ipTools Return Base IP Address = 192.168.0.1 to TC1

Step 6 - Set global variable Base IP Address = 192.168.45.62 in ipTools

Step 7 - Start TC2 from the test suites and TC2 also needs the variable value as Base IP Address

Step 8 – TC2, it Request a value as Base IP Address which also needs to get from ipTools

Step 9 - Since the global variable Base IP Address exists in ipTools so return Base IP Address = 192.168.45.62 in TC2

In this environment during the test suite execution user can change the variable value as Base IP Address. And user can put the different test cases which needs different valuables in one test suit.

Here user can set the global variable value during any test suite executions. During the test suites execution user can change this variable value or unset the variable value when TC required, through ipTools, the ipTools will always check the global variable value is exists or not, if exists then return the global variable value, if not the CETP return the default value.

So this environment will provide the flexibility of reconfiguring the test framework thus allowing a large number of automated tests to be executed sequentially in the different instruments without requiring any manual intervenor.
4.2 Scenario 2- Current structure and distribution of logic in the test environment

The current structure and distribution of logic is the result of a frequent update of the test environment. It has grown in content and is now at a size where it becomes difficult and complicated by one person to have a good overview of needed changes when there is impact. That’s why we need to improve the current structure and distribution of logic in the UP IP I&V test environment.

4.2.1 Analysis for improvement of the Current structure and distribution of logic in the test environment

Regarding the current structure and distribution of logic in the test environment, we have come up with the following implementation step for maintainability improvement of test automation environment.

- Restructure the test software iptoTools, split up the iptoTools into several groups according to different functionality and dependencies.

- Test new iptoTools with legacy or test suites. Update the new iptoTools in ClearCase.

4.2.2 Result of the Current structure and distribution of logic in the test environment

The main concept of UP IP I&V test environment behind iptoTools contains the functions implementing IP functionality and this functionality coding are increasing day by day longer and longer. For this reason it is difficult and complicated by one person to maintain or give a good overview if needed to change, so split the test software into several groups.
Figure 4-3. Different groups in ipTools

- Based on these groups, separate "old" ipTools to new ipTools. Restructure the groups to different files named by their main functionality.
- In the new test software ipTools (the name is the same as old) has its own functionality and global variables name which does not change compared with the old one.
- No impact on test cases.
- All source are stored in appropriate locations in the VOB. and location is `/vobs/cello/tiv/IP/automation/cetp/src/........`
- After splitting the test software into several groups we concatenate in one file and the location of this file is

`/vobs/cello/tiv/IP/automation/cetp/libs/ipTools.exp`

This is the combined executable new ipTools software.

4.2.2.1 Software Description

ipTools

After improvements the iptools has its own procedures and this procedure does not have any dependency with others.
TrafficGenerator

This is the common file that is used to generate the traffic, configure and send traffic using either the test instruments Smartbits or AgilentN2X. This file contains the procedures to execute test cases and test suites under ipTools. It is kept under the path -


Traffic

This is the common file that is used for traffic and kept under the path

/vobs/cello/tiv/IP/automation/cetp/src/traffic.exp.

This file contains traffic functionality with the node and using node and contains function checking on sniffer pc specified in CETP.cfg.

Mo

This is the common file that is used for all Managed Objects like creating, deleting and function for controll or handle Managed Objects on CPP node. This file kept under the path -

/vobs/cello/tiv/IP/automation/cetp/src/MO.exp

DNS

This is the common file that is used function for configure the Domain Name Server and kept under the path -

/vobs/cello/tiv/IP/automation/cetp/src/DNS.exp

Node

This is the common file that is used to connect with the nodes and function for node control, running test cases and decide node restart in previous test cases and kept under the path -

/vobs/cello/tiv/IP/automation/cetp/src/Node.exp

DLink

This is the common file that is used to connect with the DLink switches and kept under the path -

/vobs/cello/tiv/IP/automation/cetp/src/DLink.exp
Cmd

This is the common file that is used to connect and disconnect any SSH and kept under the path -

/vobs/cello/tiv/IP/automation/cetp/src/Cmd.exp
4.3 Scenario 3 - Tool environment with Model based testing

The UP IP I&V organization have today a huge number of automated test cases that cover testing of legacy functionality. For a new IP system version there could be several hundred test cases that have to be rewritten using the TCL/Expect. This result in a lot of rework. By using Model based test design, requirement changes and added new functionality can easily be introduced without to much rework of test legacy. In parallel with this thesis work a first try with Model based testing is ongoing in the organization. The UP IP I&V teams use today different configurations and different platforms both in UNIX and LINUX. Therefore we consider to improve the situation better and integrate most software’s into one system. If successful, the time will be reduced and the structure and environment will be more organic.

4.3.1 Analysis for improvement of the Tool environment with Eclipse and Model based test design

To integrate everything into one system, we first create one IDE environment to modify the test cases for regression test and function test. There are lots of IDE so it is difficult to compare one suitable IDE for UP IP I&V organization. Here are some of features of different IDE as follows:

- **Netbeans**

  The NetBeans IDE is a free and open-source Integrated Development Environment for software developers.[25] NetBeans refers to both a platform for the development of Java desktop applications, and an integrated development environment developed using the NetBeans Platform.

  The NetBeans Platform allows applications to be developed from a set of Modeler software components called modules.[28] A module is a Java archive file that contains Java classes written to interact with the NetBeans Open APIs and a manifest file that identifies it as a module. Applications built on modules can be extended by adding new modules.[28] Since modules can be developed independently, applications based on the NetBeans platform can be easily and powerfully extended by third party developers.
• **CodeGear**

CodeGear invented the first commercial Integrated Development Environment, which provides comprehensive capabilities for developers to efficiently produce software. Worldwide there are approx 3.2 million developers that use this tool to build their applications to make the digital world work. For dynamic languages such as PHP and Ruby new areas are implementing in CodeGear.

The advantage of CodeGear is that it is easy to code and requires no policies[32] or rule description classes.

• **Eclipse**

Eclipse is also free and open source community whose projects are focused on building an open development platform consisting of extensible frameworks, tools and runtimes for building, deploying and managing software across the life cycle.[29] A large and vibrant ecosystem of major technology vendors, innovative start-ups, universities, research institutions and individuals extend, complement and support the Eclipse platform.[33]

In Eclipse user can easily import their applications and software as plug-in with several methods like RCP (Rich Client Platform), DLTK (Dynamic Language Toolkit) and so on.
Users can extend Eclipse SDK\textsuperscript{37} capabilities by installing plug-ins written for the Eclipse software framework,[31] such as development toolkits for other programming languages, for instance, like C/C++, TCL, PHP, Python etc, and users can write and contribute their own plug-in modules.

One of the key benefits of the Eclipse Platform is realized by its use as an integration point. Building a tool or application on top of Eclipse Platform enables the tool or application to integrate with other tools and applications also written using the Eclipse Platform.

In UP IP I&V, testers use TCL language to develop test case, control traffic generators by ipTools library functions. The newest version of Eclipse has the TCL library for TCL designer, and it support the possibility to easily access ipTools library for tester.

Under this situation, tester needs an integrated environment which can include most testing software’s and applications. Regarding this condition, Eclipse is very powerful and good at integrating different tools and functionality.

\textsuperscript{37} SDK is a software development kit, a programming package that enables a programmer to develop applications for a specific platform.
• Regarding the advantage of Eclipse, such as open source, free to get, easily to install and learn, more compatible and powerful than other IDE platform. Compared Eclipse with Netbeans, CodeGear and other IDE Platform, Eclipse is easy to install and learn, and model base test design using RSD-RT is integrated in eclipse environment so it is more powerful than the other IDE platform. The most popular versions of two IDEs are Netbeans 5 and Eclipse 3.3 have far more similarities than differences.[24]

• But in UP IP I&V requirement we have to check in which IDE has the syntax checking, code completion, and code folding and support for Ant, CVS and JUnit and which IDE is easy to compile, run, and debug the code. Netbeans and Eclipse now have integrated GUI builders,[24] although Eclipse's is a separate component, the Visual Editor, that user must download separately.

• Many programmers prefer Eclipse[30] because of its ease of use, and because it is faster and more stable. Eclipse uses a concept called "Perspective", which is used to define multiple environments. NetBeans does not have such a concept. Eclipse support DLTK but Netbeans and CodeGear are not ready for using DLTK.

The plug-in theory in Eclipse is a lightweight software component framework, which means its architecture supports Rational Systems Developer simplifies the complexity of systems delivery. For these reason we chose Eclipse as our implementation target and regarding this advantages we can use to improve our test environment as well as serve for our Model Based Testing Design in the future.

• We integrated a web browser into Eclipse in order to let tester login into TPT and TPT2 on Eclipse without using external web browser.

• We integrated ipTools and CETP into Eclipse, so the tester can run test case and write test script in Eclipse.

• We import a verification model from current Conformiq tool into RSD-RT and get it to work properly with Qtroniq test-generator in the Eclipse environment.
4.3.2 Result of the Tool environment with Eclipse and Model based testing

The activity aims is to import a model into a RSD RT modeling tool and make it work properly together with the Qtronic test generator. RSD-RT was developed in an Eclipse environment and for dynamic language toolkits are support so we choose Eclipse to reach our final goal. Before doing the Model based test design we have to create the TCL design based on CETP and ipTools in an Eclipse environment. We use Eclipse version 3.3.1.1 and in this version already has the TCL platform in itself as a plugin; you can select for, the Eclipse Dynamic Language Toolkit – TCL Development Tools.

We use DLTK 0.9 (Dynamic Language Toolkit) and this DLTK is structural source code model. Many of JDT-alike features implemented in TCL IDE component and its launching and debugging over DBGp protocol. In Eclipse all plugins contain appropriate license files and all committer have completed Eclipse committer agreements and have been approved by the PMC(The Projects under this Charter are managed by a group known as the Project Management Committee, which called PMC).

Based on the requirements of UP IP I&V, it has huge number of test cases which are not model based, so these test cases are updated in TCL language manually every time. With implementation TCL environment in Eclipse platform, these test cases can be updated into Eclipse environment and also can be check or debug at the same time in Eclipse TCL environment.

In UP IP I&V has thousand of test cases are model based but not in RSD RT based so those model we have to import into SCTP (Stream Control Transmission Protocol) port distribution test model in the RSD RT and then generated into Qtronic test generator.

In Eclipse, after successfully installing DLTK, the user can write pure TCL scripts, debug it and run it. To step by step setup the Eclipse environment see the Appendix. For debugging the TCL file we use the another plugin named 'TCL Remote Debugging'. In this debugging plugin, the user can debug the TCL code very easily.
In UP IP I&V have more than hundred of test cases so in these test cases user can check the code and after that user can run these test cases. In Eclipse environment it has its own internal web browser, in this web browser user can open any HTML or XML files. And in this browser user can access the result files in CETP and access the Testing Performance Tools (TPT).

To integrate the ipTools and CETP into Eclipse environment user can run the test script in Smartbits and AgilentN2X, we found some limitation to run the test scripts into Eclipse environment. Eclipse does not work in the same terminal it is automatically create a new terminal. Before open Eclipse make sure that user have clear case view in the workstation. Because CETP is kept in the appropriate location of VOB.

So to integrate the CETP and ipTools in Eclipse environment if user does not have the clear case view then Eclipse can not access the VOB files. When the test case run in the test instrument Smartbits need to set the library path in the user workstation. But in test instrument AgilentN2X does not have the problem with the library path.

To integrate the CETP into Eclipse environment we use CETP as an external tools to execute the test suites. In this external tools we mention the CETP config file from the VOB location and we use argument as same as the user use to run test script.

```
/vobs/cello/tiv/IP/automation/cetp/startscript.exp -config CETP.cfg
```

In this way when the users run test script in AgilentN2X then it works properly because in AgilentN2X does not need to set the library path. But in Smartbits user must need to set the library path. For Smartbits we use the shell script to set up the library path in the user workstation from the Eclipse. In this shell script work Eclipse as a external tools. The command of this shell script file as follows:

```
#!/bin/csh
Setenv LD_LIBRARY_PATH
/vobs/cello/tiv/IP/automation/smartbits/smartlib/6.00/lib:$[LD_LIBRARY_PATH]
/vobs/cello/tiv/IP/automation/cetp/bin/CETP
/vobs/cello/tiv/IP/automation/cetp/startscript.exp -config
/vobs/cello/tiv/IP/automation/cetp/libs/CETP.cfg'
```

4.3.2.1 **Compare to Qtronic Modeler**

Rational modeling tools can generate and arrange topic diagram, browse diagram, UML diagram and so on.

The Topic diagram are used for both discovery and as visual documentation because they can be saved in the model.

Using browse diagrams to view elements in the project can help user to understand the relationships between elements in the code. Browse diagrams are not persisted by Rational Systems Developer.

The UML provides a rich notation for visualizing models, including the following key diagrams:

- Use-Case diagrams to illustrate user interactions with the system.
- Class diagram to illustrate logical structure
- Composite Structure diagrams to show the internal structure of a class or component at runtime
- Component diagrams to illustrate the organization and dependencies among Modeler parts of the system
- Deployment diagrams to show the mapping of software to hardware configurations
- Activity diagrams to illustrate flows of events
- State Machine diagrams to illustrate the series of states and object can have
- Communication diagrams to illustrate behavior in terms of how objects interact
- Sequence diagrams to illustrate behavior in terms of the sequence of interactions between objects.
Figure 4-5. UML Diagrams in Rational modeling tools[23]

- But in Qtronic Modeler user can generate and arrange only the UML diagram.

In UP IP I&V user have the existing Qtronic Modeler and Qtronic Test Generator. If UP IP I&V user wants to convert this Qtronic Model into RSD-RT, user can not import this model so user have to create the same model into RSD-RT and it gives the same output as Qtronic Modeler.

This is not a bug in RSD-RT nor in Qtronic Modeler. It is impossible to convert the model from Qtronic to RSD-RT because RSD-RT does not understand the state chart of metamodel in Qtronic Modeler. That’s why its failed to import. Qtronic Modeler uses its own which is a restricted subset of UML2, but on the other hand RSD-RT can export or import models that are using UML2.X metamodel because the XMI is an OMG standard for exchanging metadata via XML and be used for any metadata whose metamodel can be expressed in EMF, MOF and so on.
Here EMF is stands for Eclipse Modeling Framework Project and the EMF project is a modeling framework and code generation facility for building tools and other applications based on a structured data model. MOF stands for Meta Object Facility, and the MOF is an Object Management Group (OMG) standard for model driven engineering.

Because of this reason user have to create a new model in RSD-RT. For instead of RSD-RT have some benefits to create a model. The system information of the RSD-RT model generates automatically, but in Qtronic Modeler user have to write the code of the system model.

4.3.2.2 Import model Conformiq tool into RSD-RT

RSD-RT (Rational Systems Developer Rational Test RealTime) is a Cross-platform solution for component testing and runtime analysis. Its main aspect is Model-driven development for software products and systems development, and that matches our requirements perfect.

In RSD-RT user will develop two types of projects.

- **UML Projects**

  UML project is a simple project that contains a model file. User can create models in two ways: either blank or based on a Rational System Developer UML model template.[23]

- **Implementation projects**

  An implementation project can contain UML class and sequence diagrams. This projects work in Java, C, C++ and so on.[23]

On the other hand Conformiq Qtronic is a tool for automatic testing that is driven by "design models". This means that Conformiq Qtronic tests a system automatically when it is given a "design model" of the system as an input. This "design model" is a description of the intended behavior of the system on some level of abstraction. It is also correct to see it as a golden reference implementation of the system.

The main benefit of using Conformiq Qtronic is an increased product quality that is achieved by using the design model as the golden reference implementation of the system. Unlike with other testing tools tests can be automatically generated from the design model.

Qtronic can be split into single separate steps:

- First an adaptation between Qtronic and the application under test has to be created. For each tested application this has to be created only once even if updates may later be possible.
• A clear single feature is chosen for implementation.

• The feature is designed e.g. using QML and Qtronic Modeler.

• Once each feature is designed it is implemented according to its design.

• After each design implementation cycle Qtronic is used to test the conformance between the design and implementation.

Working with the RSD-RT and Qtronic Model some definition differences are as follows:

<table>
<thead>
<tr>
<th>Qtronic Modeler</th>
<th>RSD-RT</th>
</tr>
</thead>
<tbody>
<tr>
<td>System</td>
<td>Top Level “System Capsule”</td>
</tr>
<tr>
<td>Inbound</td>
<td>Inevents</td>
</tr>
<tr>
<td>Outbound</td>
<td>Outevents</td>
</tr>
<tr>
<td>Ports</td>
<td>Protocol</td>
</tr>
<tr>
<td>Data</td>
<td></td>
</tr>
<tr>
<td>Records</td>
<td>Class (Passive)</td>
</tr>
<tr>
<td>Main</td>
<td>System Structure Diagram</td>
</tr>
<tr>
<td>Classes</td>
<td>Classed</td>
</tr>
<tr>
<td>Classes that derives “State Machine”</td>
<td>Capsule</td>
</tr>
<tr>
<td>Primitive Types</td>
<td>UML Primitive Types</td>
</tr>
</tbody>
</table>

Table 4-1 Definition Differences Between Qtronic and RSD-RT

In RSD-RT automatically creates a state machine for each capsule, but as the test interface itself has no behavior. So when user create a model have to know the above things in RSD-RT and/or Qtronic Modeler.
In Ericsson, they have contract with IBM for using any Rational Tools. RSD-RT is the IBM Rational System Developer for Systems Development so in Ericsson can use RSD-RT with license key. Ericsson does not need to pay any extra money for RSD-RT license.

Create a simple model in both RSD-RT and Qtronic Modeler. After creating the RSD-RT model, export it into XMI extension because for importing of this model into Qtronic Test Generator can import XMI extension. In Qtronic test generator import this model and run into Qtronic test generator to generate the test case. In the mean time user can create the same model into Qtronic Modeler and run into Qtronic test generator to generate the test case. The output of the test case are the same.

So in RSD-RT model user can export it into Qtronic test generator to generate the test case. But problem is when user create a model in Qtronic Modeler and wants to import this model into RSD-RT. User can import only the code not the whole model. The user have to create the same model into RSD-RT.

Compare to RSD-RT and Qtronic, RSD-RT user can create a model not to generate the model. For generating the model user need test generator that is provided Qtronic. In Ericsson, to generate the various test cases UP IP I&V uses the Qtronic Modeler and Qtronic Test Generator but in other I&V user use the RSD-RT Modeler, so in future UP IP I&V user need to import the RSD-RT model.

Qtronic Modeler works for only the state machine diagrams so user can create a model very easily and its take time for one model 10 minutes which you can see in Figure 4-6, but it depend on the user to create a model. In the same model in RSD-RT takes time minimum 30 minutes, that means to draw a model in RSD-RT instead of Qtronic multiple by 3 times. But user can save time from the coding part. Before we discuss that RSD-RT automatically generate the system information when the user create a model.
In Figure 5-6 shows a simple state machine diagram in Qtronic Modeler and this state machine diagram works to set the port range size. First it wait for the port range size and after getting the size of the port then it set. This simple state machine diagram if user wants to create into RSD-RT then in RSD-RT user have to create a system capsule for the model and in system capsule needs to create a system boundary protocol and this protocol user need to create an inevent and outevent. Figure 5-7 shows that the system boundary in RSD-RT.
Figure 4-7 RSD-RT Module in System Boundary

In input and output events we have to define data and this is done by using classes because in RSD-RT classes that have no behavior in the diagram. In RSD-RT classes have to create the attribute and in this attribute user have to define the value type like string or integer and make it public visibility. In system boundary protocol and top level of the system make the connection between them. See Figure 5-8. After connection to the system it's automatically created the state machine diagram. But in this state machine diagram only create the initial state and a basic state. So make a new state and make a transition from the basic state that is already in the state machine to state created.
Figure 4-8 Connection between system and system boundary in RSD-RT

Previous we describe that in RSD-RT user can import only the code from Qtronic Modeler, so after make the transition between the two state in code view part user can copy the code or write the desire code. Figure 5-9 shows the transition between the two state.
In our model we have two attributes in the class one is inEvent and one is outEvent so user have to import the code for both attributes. After finish the code change the implementation language of the action from C++ (RSD-RT default) to QML.

In our model to create in Qtronic it takes time 10 minutes but when we created the same model in RSD-RT it takes 30 minutes. Table 4-2 show the time difference between two Modeler when we create the same model and Figure 5-10 shows that the time average chart when we

<table>
<thead>
<tr>
<th>Modeler</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Qtronic Modeler</td>
<td>10</td>
</tr>
<tr>
<td>RSD-RT</td>
<td>30</td>
</tr>
</tbody>
</table>

Table 4-2 Time difference to create a model
create the same model in Qtronic Modeler and RSD-RT. Figure 5-10 shows time average only for draw the model but considering the coding part we discuss before RSD-RT create the system automatically so user do not need to write system function code but in Qtronic user need to declare the system function code. So here RSD-RT user save some time to create a full model.

This result is save lot of rework by using Eclipse environment and using the model base test design can easily be executed without to much rework of the test legacy. We suggest to improve the future work in different platforms both in Linux and Unix, so after that improve section 7.3 the structure of UP IP I&V work environment will be more organic.
4.4 **Guideline for system developers & the tester**

This section is useful for developers who want to add or modify procedures in ipTools. And who want to add or modify new test cases in Eclipse environment.

**Adding new procedures or modify the old procedures**

This sub section describes the guidelines for adding new procedures or modify the old procedures in ipTools. User define the procedures name that will added and to know how this procedures works for and clear concept about ipTools.

- First user must have the clear case view because ipTools are kept in the appropriate VOB locations.
- User can checkout the appropriate file with reserved from the VOB
- Add the new procedures and/or modify the previous procedures
- After modifying or adding the procedure user must concatenate the file in clear case
- Check in the file with appropriate VOB location

**Model test designer**

This sub section describe the guidelines for export RSD-RT model into Qtronic Modeler and vice versa in Model based testing environment.

- First user must have the clear case view and in the config spec write the following lines for use RSD-RT

  ```
  element * RSDRT_I8-5
  element /vobs/cello/cade_struct/setup/license/... /main/LATEST
  ```

- User must setup the Qtronic test generator and Modeler also.
- In RSD-RT export the model in XMI extension
- For Qtronic test generator create one QML file and inside this file user write the exported file name, for example: filename.xmi
Tester

This sub section describes the guidelines for the tester who want to run test cases or modify test cases in Eclipse environment.

- Make sure that user have the appropriate Eclipse environment. If not see Appendixes.
- Define the name of the file that will execute the test case in Eclipse environment. The name of the file should be filename.tcl
- Save this file in the home directory
- Before run the test cases restart the node
- For run test cases in Eclipse environment use CETP as a external tools
- In the argument list give the appropriate location of the test case TCL file with the CETP config command
- In Linux platform user can not run test cases in Smartbits
- In Linux platform make sure that user have locally install in MoShell and change the location of the MoShell in CETP config file
- In Unix workstation for running test case in Smartbits does work then user should use shell script (see section 1.15)
5 Limitation of the Thesis Work

After doing our thesis work test environment is not fully integrated because of time limitation and not so much related work we can use and consult for our target. During the thesis work we found some limitation. These are as follows:

- In clear case view of Ericsson, the library file of the programming language is not updated frequently so the version of the programming language is older. For this reason, we face the problem to work.

- In eclipse environment user want to use the testing performances tools but in Unix platform we face the problem to use the internal web browser because the testing performance tools is in the internal web address. So we gave the solution for Unix platform how to use the external browser in Eclipse environment but in Linux platform it is possible to use the internal web browser into Eclipse environment.

- Eclipse does not use the same terminal when user run the test cases, internally eclipse create another terminal to run the test cases so test cases does not find the CETP file to access the test cases.

- In UP IP I&V organization has the test instrument Smartbits. To run the test cases into Smartbits it is impossible in Linux platform now a days. We suggest to improve Linux platform for Smartbits as our future work.

- In Ericsson has their own shell named MoShell and its located in the appropriated location of VOBS, so for Unix user can access this shell but the Linux user can not access this so for Linux user can setup the MoShell in the local directory and mention it in the CETP config file.

- In Model base test design for RSD-RT support only TCShell not any other shell like bash shell. So we suggest to improve in our future work.

- Create a model into RSD RT for Linux user have to source the setup file but for Unix user does not need to source the file. The setup file is located in the appropriated location of the VOBS.
6 Conclusion

In this thesis work all the major goals have been achieved which were set in the beginning of the work. The goal of this thesis was to write a report on flexibility and maintainability of automation test environment. We implemented the start variable for running the different test cases and the test software modify or updated separately and then concatenate all separated files into one file. We introduced the new environment of modifying the test cases and finally we convert the SCTP port distribution test model from conformiq Qtronic to IBM environment (RSD-RT). All the results and measurements are quite satisfactory and have fulfilled the major goals of this thesis work.

The thesis results shown that the Eclipse can be used as a IDE for TCL and run the test case working with CETP and ipTools and to use the test performance tools to import the test case results. Our results found RSD-RT is a very effective software development in real time and it can be very efficient to create a SCTP port distribution test model. To import the model Qtronic to RSD-RT, our thesis result found this is not a bug to UML.

6.1 Problems

While performing the thesis work we encountered a number of problems. Each of these problems taught us valuable lessons from different perspectives. Here we present a few of those problems:

We had way too much information about the different component of CPP when we started working. The Ericsson databases have a comprehensive collection of documents describing various aspects of CPP. At first we were really clueless about what to study and what to ignore. In this process we wasted a lot of valuable time reading through documents that we later found out to be irrelevant to our work.

When we started working in the test laboratory at IP I&V we had absolutely no idea about the test automation process. We did not even have the clue on how to handle the test instruments. To make the matter worse there were no documentation available that was aimed at a layman user. So we had to plough through the Ericsson databases for clues. Sometimes other testers at the laboratory had to waste their valuable time to answer our most fundamental queries.
### 6.2 Lessons Learned

We learned a lot of valuable lessons while performing the thesis work. In this section we would like to point out a few of those.

We learned a great deal about CPP and its components. CPP is one of Ericsson's major platforms. A large number of its 3G specific developments are based on CPP. So it was a great opportunity to learn about the platform.

We learned a lot about test automation process. It was a very good opportunity for us to exercise our scripting skills.

It was a nice opportunity to work in a professional setting which is so much different that the academic setting.

The thesis work was a true test of team work. It required the whole hearted efforts of both the members to make this thesis work a success.

### 6.3 Future Work

The following problems has been indentified and need to further investigation. This thesis work also had some constraints which can be considered in the future experiments.

Due to the time and some particular equipment limitation we could not improve the whole test environment but still can do more improvement in the test environment.

When tester run lots of test cases in the night, some test case passed, some test case failed, it’s very normal situation. But a simply problem happen frequently, for example, when tester run hundred of test cases for fifty times, some test cases (we can say about like 10 test cases) always failed in the first twenty times or thirty times, but pass in the next ten times and twenty times. Then many tester consider these problems as instability in the test environment, and we propose to improve the stability of the test environment if possible.

In UP IP I&V organization has the huge number of test cases and these test cases run in the test instrument either AgilentN2X or Smartbits or both. But working in Linux platform it is not possible to run the test cases in Smartbits. So we propose to improve Linux platform environment for the tester.

Working with the model base test design for RSD-RT support only TCShell not any other shell like bash shell. To run RSD-RT user face this problem frequently so we suggest to improve the RSD-RT environment.
7 Abbreviations

3G 3rd Generation
3PP Third Party Products
AAL2 ATM Adaptation Layer type 2. Multiplexing of multiple
AAL2U streams into one path for efficient bandwidth
utilization.
API Application Programming Interface
ATM Asynchronous Transfer Mode
BP Board Processor
CBU Control Base Unit
CELLO see CPP
CETP Common Expect Test Platform
CPP (Ericsson) Connectivity Packet Platform
CV Configuration Version
CVS Configuration Version Specification
DBGp Debugging Protocol
DBM Database Management
DLTK Dynamic Language Toolkit
DNS Domain Name System
DU Design Unit
EMF Eclipse Modeling Framework Project
ET Exchange Terminal
ET-MFG Extension Terminal Multi Function Gigabit
FT Function Test
GPB  General purpose Processor Board
GUI  Graphical User Interface
HTML  Hyper Text Markup Language
HW  Hardware
I&V  Integration and Verification
IP  Internet Protocol
ITU  International Telecommunication Union
JDT  Java Development Tool
MO  Managed Object
MOM  Managed Object Model
MFG  Multi Function Gigabit
MFX  Multi Function Switch
MOF  Microsoft Operation Framework
MP  Main Processor
NTP  Network Time Protocol
OS  Operating System
PC  Personal Computer
PMC  
RBS  Radio Base Station
RNC  Radio Network Control
RTCP  Real Time Control Protocol
RTP  Real Time Protocol
SCTP  Stream Control Transport Protocol
SDK  Software Development Kit
SPB  Special purpose Processor Board
<table>
<thead>
<tr>
<th>SSH</th>
<th>Secure Shell</th>
</tr>
</thead>
<tbody>
<tr>
<td>TUB</td>
<td>Timing Unit Board</td>
</tr>
<tr>
<td>TC</td>
<td>Test Case</td>
</tr>
<tr>
<td>VOB</td>
<td>Versioned Object Base</td>
</tr>
<tr>
<td>VS</td>
<td>Verification Specification</td>
</tr>
<tr>
<td>WS</td>
<td>Workstation</td>
</tr>
<tr>
<td>XMI</td>
<td>XML Metadata Interchange</td>
</tr>
<tr>
<td>XML</td>
<td>Extensible Markup Language</td>
</tr>
</tbody>
</table>
8 References


   US&1c=eng&cmpid=4533


   EAod-i4r6q

   what_is_uml.htm


   eecnbeans/


     m1#q1


eclipsecommunity.php

Appendixes - A

########################################################################

# PROCEDURE: ::ipTools::getCfg

#

# DESCRIPTION:

#     Returns a configuration value previously read from the config file
#
#     otherwise it returns a configuration value previously set from the
#
#     global variable.
#

# ARGUMENTS:

#     aCfgName - Name of config variable.
#

# RETURNS:

#     Returns the Value of the Variable.
#

# USAGE:

#     -
#

# SEE ALSO:

#     ::ipTools::setCfg, ::ipTools::unSetCfg

########################################################################
proc ::ipTools::getCfg { aCfgName } {
    ::cetp::TRACE ENTER \"$::ipDefs::INFO_COLS\" \n    \"args : $aCfgName\" \n    \"$::ipDefs::DTTERM_NORMAL\"
    if { [info exists gCfgArray($aCfgName)] } {
        set lreturnval $::gCfgArray($aCfgName)
    } else {
        set lreturnval [::cetp::getCfg $aCfgName]
    }
    ::cetp::TRACE RETURN \"$::ipDefs::INFO_COLS\" \n    \"RETURNS ... $lreturnval\" \n    \"$::ipDefs::DTTERM_NORMAL\"
    return $lreturnval
}

# PROCEDURE: ::ipTools::setCfg
#
# DESCRIPTION:
#     It sets the value that are globally exists .
#
# ARGUMENTS:
#    aCfgName   - Name of the Variable.
#    aCfgValue - Value to set.
# RETURNS:
#
#
# USAGE:
#
#
# SEE ALSO:
#
## ::ipTools::getCfg, ::ipTools::unSetCfg

#########################################################
proc ::ipTools::setCfg { aCfgName aCfgValue } {
    ::cetp::TRACE ENTER \"$::ipDefs::INFO_COLS" \"args : \$aCfgName, \$aCfgValue\" \"$::ipDefs::DTTERM_NORMAL" 
    set lNewCfgVar [::cetp::getCfg $aCfgName 1]
    if { $lNewCfgVar == "" } {
        ::cetp::PAL "$::ipDefs::ERROR_COLS The variable \"\$aCfgName\" does not exist in the config-file! \$::ipDefs::DTTERM_NORMAL" 
        exit
    } else {
        set ::gCfgArray($aCfgName) $aCfgValue
    } 
    ::cetp::TRACE RETURN \"$::ipDefs::INFO_COLS" \"
"RETURNS ..." \\
"$::ipDefs::DTTERM_NORMAL"
}

# PROCEDURE: ::ipTools::unSetCfg
#
# DESCRIPTION:
# It unset the value that are set before .
#
# ARGUMENTS:
# aCfgName - Name of the Variable.
#
# RETURNS:
# -
#
# USAGE:
# -
#
# SEE ALSO:
# ::ipTools::getCfg, ::ipTools::setCfg

proc ::ipTools::unSetCfg { aCfgName } {
    ::cetp::TRACE ENTER \"$::ipDefs::INFO_COLS" \
"args : $aCfgName"

"$::ipDefs::DTTERM_NORMAL"

unset ::gCfgArray($aCfgName)

::cetp::TRACE RETURN \n
"$::ipDefs::INFO_COLS"

"RETURNS ..."

"$::ipDefs::DTTERM_NORMAL"

}
Appendixes - B

Installation Eclipse Environment

This section describe the several steps to install the Eclipse environment for UP IP I&V organization. This is called Eclipse installation user guide. Because in this section we describe how to install Eclipse, modify the test cases and run the test cases for the function and regression test in different platforms both in UNIX and LINUX.

Ericsson's have their Eclipse version which is called E4E (Eclipse for Ericsson). Users can download and install E4E in their workstation or users can download Eclipse from the Eclipse website and run it into their workstation. Before users install the E4E, make sure user have enough space in home drive. E4E can be installed using the IBM Installation Manager. If the user does not have IBM IM download it from the link provided below and follow the instruction in this link.

http://eclipse.internal.ericsson.com/E4E

Here user can find the IBM IM installation instruction. After successfully installing IBM IM the user can upgrade and modify E4E.

If user do not want to set up the E4E then they need to download Eclipse from the link below and follow the instruction below


After downloading Eclipse you have to unzit it and run it in your workstation. Before running make sure that you have the clear case view and run it in a shell or terminal window.

When you run the Eclipse command you have to fixed your workspaces location. After run the Eclipse it looks as follows
Now in your workstation you has Eclipse installed. Now we have to configure it first.

click on window --> Preferences
Then in the text field type proxy and select network connection
Select Manual proxy configuration and in type

HTTP proxy -------- www-proxy.ericsson.se and port 8080

SSL proxy --------- www-proxy.ericsson.se and port 8080

and then click OK.

Now you have to click on Help --> Software’s Updates --> Find and Installation

Select search for new features install and then click next
Then Select Dynamic Language Toolkit Update and click finish
After clicking finish button it will updated automatically and takes time.

After successfully installing DLTK you can run TCL in Eclipse environment.

or

you can download DLTK plugging from the web link below

http://www.eclipse.org/dltk/

and after downloading this plugging copy and paste it in your appropriate Eclipse location.

Now in your Eclipse environment DLTK plugging installed properly.

To configure the DLTK click on windows --> Preferences
Now expand TCL icon and select interpreters

in the right side of the box we will find the search button click this button and wait. It will search automatically in your TCL interpreters and then select tclsh8.4 and click ok.

For debugging we have to download the debug tools from the link below


save those tools in your workstation and unzip it. Here we use version 3.5
After download those tools we have to click again in windows --> Preferences and then expand TCL and then select Debug.
Here you give the appropriate path where you download the Debug Tool and then click Apply and ok.

Now you have to create a TCL file and Debug it and Run it as a TCL scripts.

Integrated CETP

To integrate CETP we use as external tools. So click Run --> External Tools --> Open External Tools Dialog.

After clicking open external tools dialog, this will open a box and then expand program to build a program.

In this New program give the name CETP. In the location box give the appropriate location of CETP for example:
/vob/cello/civ/test_utilities/CETP/bin/CETP
and then in Working Directory give the location of CETP config file, that could be in your home directory or in VOB. For example:

/home/<userid>/CETP
/vobs/cello/tiv/IP/automation/cetp/libs – This is VOB

In the Argument field write the command when you run a test cases. For example:

/vobs/cello/tiv/IP/automation/cetp/startscript.exp -config CETP.cfg

After giving the argument you should click Apply and then Run.

There is another way to run CETP as a external tools with the library path.

click on Run --> External Tools --> Open External Tools Dialog, create a new program named Env. Before creating a new program you have to create a text file in your home directory and in this file write the following lines.

#!/bin/csh

setenv LD_LIBRARY_PATH

/vobs/cello/tiv/IP/automation/smartbits/smartlib/6.00/lib:${LD_LIBRARY_PATH}
save this file in your home directory.

Now in Eclipse you already create a new program called Env. In this Env program write in the location field where your Env file is exists. And in working directory write the location of your CETP config file.

Now click Apply and then Run. We will see CETP is running after a while.
Integrated Web Browser

To accessing a web browser, in Linux platform no need to create web browser because in Eclipse has the internal browser so you can open in any HTML or XML file. But for Unix platform we are using external browser. But in both platform accessing TPT and CETP result file we create one program name TPT2. To create this program click on Run --> External Tools --> Open External Tools Dialog and give the name TPT2. Now in the location field write your browser location. For example:

```
/usr/bin/firefox
```

And in the Argument field write the following address,

```
file://home/esharay/CETP/results/ (Give your CETP result file location)
http://eed.ericsson.se/TPT/index.jsp
http://eed.ericsson.se/cgi-bin/tpt/tpt.pl
```

And then click Apply and Run. Now we see your firefox web browser is already open with these addresses.
Appendixes - C

Test Script example:

#########################################################################
##
## FILE: TC-IPBSA-C52.tcl
##
## Title: Set Up Session - UDP
##
## Tag: TC-IPBSA-C52
##
## Description: Verify that a complete UDP session can be set up in one
## step and that the confirm signal contains the correct
## information.
##
## Deviations: In Result2 after Action1 in TC it says that all parameters
## in confirm signal are checked and correct. The parameters
## will be checked manually.
##
## Test Setup: 1, TCS 1/174 25-CRX 101 10/1 Rev B
##
## Exec Type: Semi
##
## COPYRIGHT:
## Copyright © Ericsson AB 2006 All rights reserved. The information in this
## document is the property of Ericsson. Except as specifically authorized in
## writing by Ericsson, the receiver of this document shall keep the information
## contained herein confidential and shall protect the same in whole or in part
## from disclosure and dissemination to third parties.
## Disclosure and disseminations to the receiver's employees shall only be made
# on a strict need to know basis.
#
#
# IDENTIFICATION:
#
# @(#) ClearCase ID: /vobs/cello/tiv/IP/test/sys_func/ipbsa/TC-IPBSA-C52.tcl /main/3 07-03-21 15:26 uabstg #
#
#
# REVISION HISTORY:
#
# Date    Sign       Rev         Comment
#========================================================================
# 060706 uabstom   /main/1    STATUS:EXECUTABLE
# 060816 etxamer   /main/2    Updated for ipTools ver. 95
# 070321 uabstg    /main/3    Updated for WP80_LSV20, special fix for var
#                               IpEtEthermetId due iptools
#
#**************************************************************************
##

proc TC-IPBSA-C52 { } {

**************************************************************************

# Init of variables

**************************************************************************

# Misc variables

# ----------------------------------------

match_max 100000
set lIpEtType        [lindex ::cetp::getCfg IPET1TYPE] $::ipDefs::NODE1]
set lIpEtSubrack     [lindex ::cetp::getCfg IPET1SUBRACK] $::ipDefs::NODE1]
set lIpEt2Slot       [lindex ::cetp::getCfg IPET2SLOT] $::ipDefs::NODE1]
set lIpEt2Apn                [::ipTools::getApn
             NODE1] "$lIpEtSubrack \ $IpEt2Slot]
set lIpEt2Piu                 [::ipTools::getpiuid NODE1 $lIpEt2Apn]

set IlpTestCpSubrack [lindex ::cetp::getCfg IPTESTCP1SUBRACK]
$::ipDefs::NODE1]
set IlpTestCp1Slot     [lindex ::cetp::getCfg IPTESTCP1SLOT] $::ipDefs::NODE1]
set IlpTestCp1Apn        [::ipTools::getApn
             NODE1] "$lIpTestCpSubrack \ $lIpTestCp1Slot]

set IlpTestDpSubrack  [lindex ::cetp::getCfg IPTESTDP1SUBRACK]
$::ipDefs::NODE1]
set IlpTestDpSlot      [lindex ::cetp::getCfg IPTESTDP1SLOT] $::ipDefs::NODE1]
set IlpTestDp1Apn        [::ipTools::getApn \ $lIpTestDpSubrack \ $lIpTestDpSlot

set lBaseIpAddr  [::cetp::getCfg BASE_IP_ADDRESS]

# Variables for IpEtEthernet
#---------------------------------------------------------------

set lIpEtEthernetParent    [::ipTools::getParentLdn \

NODE 1 \
"IpEtEthernet" \\
lIpEtSubrack \\
lIpEt2Slot]

set lIpEtEthernetId [expr ($::ipDefs::IPET_ETHERNET + 1)]

# Variables for IpInterface

# -------------------------------------------------------------------

set lIpInterfaceId $::ipDefs::IP_INTERFACE

set lBaseIpAddr [::cetp::getCfg BASE_IP_ADDRESS ]

set lIpLsos 0.1

set lDefaultRouter0 [::ipTools::newIpAddr $lBaseIpAddr $lIpLsos]

set lDefaultRouter0 [::ipTools::newIpAddr $lBaseIpAddr $lIpLsos]

# Variables for IpAccessHostEt

# -------------------------------------------------------------------

set lIpAccessHostEtParent [::ipTools::getLdn NODE 1 ]

set lIpAccessHostEtParent [::ipTools::getLdn NODE 1 ]

"IpAccessHostEt"

lIpEtSubrack

lIpEt2Slot
set lIpAccessHostEtId $::ipDefs::IPACCESS_HOST_ET

set IPlugInUnitMoRef [:ipTools::getMoLdn NODE1 "PlugInUnit"

$lIpEtSubrack

$lIpEt2Slot ]

set lIpLsos 0.101

set lHostEtIpAddress [:ipTools::newIpAddr $lBaseIpAddr $lIpLsos]

# Variables for Traffic Generator

# -------------------------------------------------------------------

set ITgIpAddress [:cetp::getCfg TRAFFICGENIP ]

set ITrafficGenerator [:cetp::getCfg TRAFFICGEN ]

set ITgSlot [:cetp::getCfg TRAFFICGENSLOT ]

set ITgPort [:cetp::getCfg TRAFFICGENPORT ]

set INoOfStreams 1

set IProtocol "UDP"
**# Precondition**

::cetp::PAL "=================================================================================================="

::cetp::PAL " Create IpEtEthernet MO "

::cetp::PAL "=================================================================================================="

set lIpEtEthernetMo [::ipTools::createIpEtEthernetMo \ 
  MOSHELL1 \ 
  $lIpEtEthernetParent \ 
  $lIpEtEthernetId]
::cetp::PAL "======================================================="
::cetp::PAL " Create IpInterface MO    "
::cetp::PAL "======================================================="

set $lIpInterfaceMo [::ipTools::createIpInterfaceMo $MO SHELL 1
[index
$lIpInterfaceId
$lDefaultRouter0

::cetp::PAL "======================================================="
::cetp::PAL " Create IpAccessHostEt MO   "
::cetp::PAL "======================================================="

set $lIpAccessHostEtMo [::ipTools::createIpAccessHostEtMo $MO SHELL 1
[lIpAccessHostEtParent
[lIpAccessHostEtId
[lIpInterfaceMo
[lHostEtIpAddress
[lPlugInUnitMoRef

::cetp::PAL "======================================================="
::cetp::PAL " Initiate ipapplregi, ipapplsci and spm    "
::cetp::PAL "======================================================="

::cetp::cmdTarget NODE1 "lhsh $lIpTestCp1Apn iptest initiate lists"
::ipTools::iptest NODE1 initiate -iptestcpapn $IpTestCp1Apn -ipetapn $IpEt2Apn -iptestdpapn $IpTestDp1Apn

::cetp::PAL "=======================================================
::cetp::PAL " Enable traces in application
::cetp::PAL "=======================================================

::cetp::cmdTarget NODE1 "lhsh $IpTestCp1Apn te enable rec_sig iptestapp_cp_proc"

::cetp::PAL "=======================================================
::cetp::PAL " Check for ERRORS
::cetp::PAL "=======================================================

set ITELogErrorList [::cetp::getTeLogErrors NODE1 "all"
$IpTestCp1Apn]  
set CompareErrorList [::cetp::compareTeLogErrors ITELogErrorList]

if {([llength CompareErrorList] == 0)} { 
  ::cetp::PAL " NO ERRORS FOUND "
} else {
  ::cetp::cmdReport 
  "Errors found"
  ==
  ==
  $::cetp::REPORT_CMD_NOK
  ::cetp::setTCState $::cetp::VERDICT_FAILED
}

### Clear T&E log

::cetp::cmdTarget NODE1 "lhsh $IpTestCp1Apn te log clear"
### Action

::cetp::PAL "=================================================================================
::cetp::PAL " Set up one UDP session
::cetp::PAL "=================================================================================

::cetp::cmdExpect "DONE"

if {::cetp::cmdTarget NODE1 "lhsh $lIpTestCp1Apn iptest setup -remoteIpAddress $lTgSourceIpAddress -remoteUdpPort $lSourcePort -localUdpPort $lLocalUdpPort" != ::cetp::CETP_OK} {
    ::cetp::cmdReport
    "Could not set up a UDP session"
    ""
    "$::cetp::REPORT_CMD_NOK"
    ::cetp::setTCState ::cetp::VERDICT_FAILED
}

### Result

### Verify that application receives confirmation that the session is set up

::cetp::PAL "=================================================================================
::cetp::PAL " Appl. receives a confirmation that session is set up"
::cetp::PAL "=================================================================================

::cetp::cmdExpect ".*CELLO_IPAPPLSCI\[0-9\]*_SESSION_SET_UP_CFM"

::cetp::cmdExpect ".*CELLO_IPAPPLSCI\[0-9\]*_SESSION_REQUEST_OK"

if {
    ::cetp::cmdTarget NODE1 "lhsh $lIpTestCp1Apn te log read" != ::cetp::CETP_OK
} {
    ::cetp::cmdReport "UDP session not set up properly"

    ::cetp::setTCState ::cetp::VERDICT_FAILED

} 

###########################################################################
# Result
###########################################################################

### Verify the parameters in the session set up confirm signal
::cetp::PAL "======================================================="
::cetp::PAL "Verify all the parameters above manually, are they correct?"
::cetp::PAL "======================================================="

::ipTools::pauseScript

###########################################################################
# Result
###########################################################################

### Verify that application receives confirmation from ANS that AAL2 connections has been created
::cetp::PAL "================================================================="
::cetp::PAL "==============================================="
::cetp::PAL " Verify the AAL2 connection          "
::cetp::PAL "==============================================="

::cetp::cmdExpect ".*CELLO_AAL2NCI_CONN_CFM"

if { [::cetp::cmdTarget NODE1 "lhsh $lIpTestCp1Apn te log read"] != $::cetp::CETP_OK } {
    ::cetp::cmdReport
        "AAL2 connection not set up properly"
    ::cetp::setTCState $::cetp::VERDICT_FAILED
} }
::cetp::PAL " Create a UDP stream "
::cetp::PAL "=================================================================
::ipTools::tgaddStreams
$lHostIpAddress
$lTgSourceIpAddress
$lNoOfStreams
$lProtocol
$lPacketLength
$lProtocolFields
::cetp::PAL "=================================================================
::cetp::PAL " Send continuous traffic "
::cetp::PAL "=================================================================
::ipTools::tgsendPackets
$lPktRate
$lNoOfPkts
::ipTools::mySleep 60
::cetp::PAL "=================================================================
::cetp::PAL " Stop the traffic "
::cetp::PAL "================================================================="
::ipTools::tgstopTraffic

# Result
::cetp::PAL "###########################################################################
::cetp::PAL "::cetp::PAL "Verify sent packets = receive packets"
::cetp::PAL "###########################################################################
if { [ ::ipTools::matchTgSentTgRec ] != $::ipDefs::OK } {
   ::cetp::cmdReport
   "Mismatch between RX and TX counters!"
   ""
   "$::cetp::REPORT_CMD_NOK
   ::cetp::setTCState $::cetp::VERDICT_FAILED
}

###########################################################################
# Restore
###########################################################################
::cetp::PAL "###########################################################################
::cetp::PAL "Disconnect to traffic generator"
::cetp::PAL "###########################################################################
::ipTools::tgdisconnect
::cetp::cmdExpect "DONE"
if { $::cetp::cmdTarget NODE1 "lhsh $lIpTestCp1Apn iptest release all" != $::cetp::CETP_OK } {
    ::cetp::cmdReport
    "Could not release UDP session"
    \
    ""
    "$::cetp::REPORT_CMD_NOK"
    ::cetp::setTCState $::cetp::VERDICT_FAILED
}

::cetp::PAL "=========================================================================="
::cetp::PAL " Disable traces in application "
::cetp::PAL "=========================================================================="

::cetp::cmdTarget NODE1 "lhsh $lIpTestCp1Apn te disable rec_sig iptestapp_cp_proc"

::cetp::PAL "=========================================================================="
::cetp::PAL " Terminate ipapplregi/ipapplsci "
::cetp::PAL "=========================================================================="

::cetp::cmdTarget NODE1 "lhsh $lIpTestCp1Apn iptest terminate all"

::cetp::PAL "=========================================================================="
::cetp::PAL " Remove MOs and check errors "
::cetp::PAL "=========================================================================="

::ipTools::postCmd

### Set TC state to PASSED

::cetp::setTCState $::cetp::VERDICT_PASSED
<table>
<thead>
<tr>
<th>Prepared (also subject responsible if other)</th>
<th>No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>/EAB/FTP/DNF   ESHARAY EHANJIN</td>
<td>/EAB/FTP/DNF- Uen</td>
</tr>
<tr>
<td>Approved</td>
<td>Checked</td>
</tr>
<tr>
<td>/EAB/FTP/DNF Jan Å Eriksson</td>
<td>Date</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

}