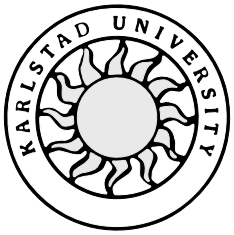

Computer Science

Henrik Bergkvist

Nicklas Jansson

Implementation of

JAIN MAP API CTS

Bachelor's Project

2000:13

Java Application Programmers Interface

Integrated Network

Mobile Application Part

Application Programmers Interface

Conformance Test Suite

JAIN MAP API CTS

Henrik Bergkvist

Nicklas Jansson

This report is submitted in partial fulfillment of the requirements for the Bachelor's degree in Computer Science. All material in this report which is not our own work has been identified and no material is included for which a degree has previously been conferred.

Nicklas Jansson

Henrik Bergkvist

Approved: 2000-05-13

Advisor: Nils Dåverhög

Examiner: Stefan Lindskog

Abstract

This document describes the test application for the JAIN MAP API standardization project. The description of the application and a description of the project as well as how the project proceeded are also included in this report.

The background of the JAIN MAP API is described as well as all the problems that were encountered during this project.

The main application for the CTS is described in words and pictures how it works, and also how it was implemented and tested.

An evaluation of all the programs that was used is also included.

Sammanfattning

Detta dokument beskriver testapplikationen till standardiseringsprojektet av JAIN MAP API.
En beskrivning av applikationen och projektet samt hur projektet har genomförts är också inkluderat i detta dokument.
Bakgrunden till JAIN MAP API beskrivs samt problemen som uppkom under projektets gång.
Huvudapplikationen för CTS är beskrivet i ord och bild hur den fungerar, och också hur den implementerats och testats.
En utvärdering av alla program som användes finns också inkluderad i detta dokument.

Contents

List of Figures

1 Introduction

As one of the last courses in acquiring a Bachelors Degree in Computer Science at the University of Karlstad a student has to work as a part of a project at a computer related company or at a research project at the university. One of the goals of the course is to produce this document among other things. Our project was to construct a test tool for Ericsson Infotech to test a standardization of an Application Programmers Interface for the Mobile Application Part. Our supervisor at Ericsson Infotech was Sören Torstensson, who has been an asset to us during the project and helped us when we were lost.

Here in this document we will try present our work and how we proceeded during the project.

First we will introduce a background, where we will cover some of the reasons why we are implementing a conformance test suite for the JAIN MAP API and why the API in the first place is implemented. After the background there will be a section explaining and discussing the problems we faced during the project and how working for a large corporation effected our job. Following the problems we will describe the goals of the project as well as the goal of our application.

In chapter three the prerequisites and demands of Ericsson will be discussed, what they are and why they have them.

The next chapter describes how we constructed the application and how all the pieces of the JAIN MAP project fits together.

Chapter five describes how we did the implementation and how the testing proceeded. The implementation section describes which programs we used for what part of the implementation. The testing section describes in more detail how we did the testing and what we tested.

In the "Experience and recommendations" chapter we introduce the experience gained during the project and also a more extensive introduction to the programs we used, also what we recommend when testing an application.

2 Background, problems and goals

In this part we present the background for the project, that is why this project was started to start with and what the companies involved have to gain from it.

We will also present the problems we encountered during our implementation of the application and how we solved them to some degree. Some of the problems we had will most likely apply to many others that are involved in similar projects at similar corporations.

The goal of the project will be presented to the reader as far as we could comprehend, because it is a somewhat complex issue that is difficult to explain since we was not supposed to implement the full API for the JAIN MAP API standardization project.

2.1 Background

With the more frequent use of the different services of the mobile telephone, Sun Microsystems, APiON Ltd., Ulticom, ADC Newnet, Ericsson, Telcordia, Trillium Digital Systems and NTT decided to develop an Application Programmer Interface (API) standard for the Mobile Application Part (MAP) which uses the Signaling System 7 (SS7) protocol. The SS7 protocol, which previously was accessed in a more complicated matter, contains a multitude of parameters and variables that the application programmer has little or no interest in defining. The API works as an interface between the SS7 stack and the application, this will remove much of the complexity of implementing applications directly onto the SS7 stack.

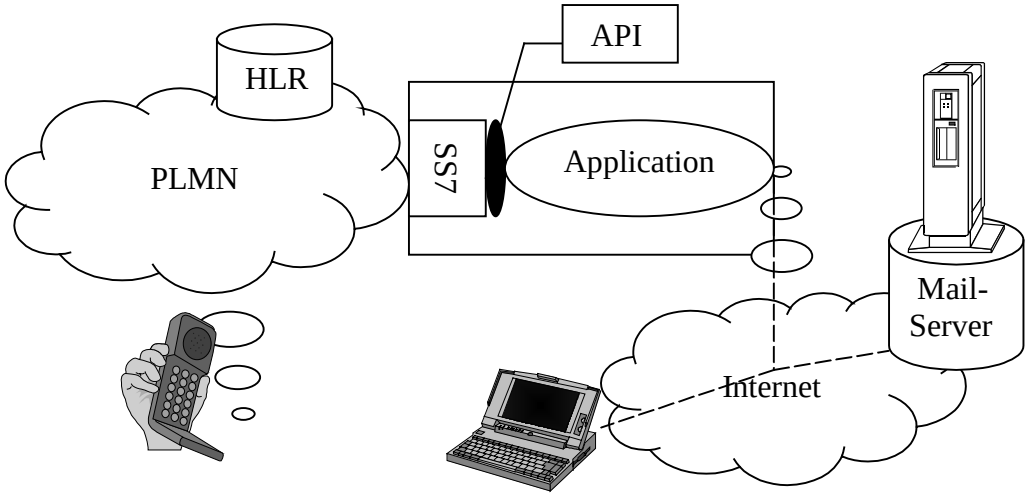

Figure 2.1: API Description

Previous and similar projects have taken place between the different companies, one in particular has been a source for ideas for this project, named JAIN TCAP API standardization. The TCAP API deals with stationary phones and not mobile phones as the MAP API project does. The TCAP API was developed at Sun Microsystems, as far as we know. The MAP API standard is being developed partially at Ericsson Infotech, while a parallel development is taking place at the other companies. Developing a standard at different companies at the same time requires a lot of contact between the software designers so that everyone is aware of how it should work and look.

The reason for developing this API is to make it easier for the mobile phone service providers to develop their own specific services. Making it easier for the service providers will increase the available services and hopefully produce totally new and unheard of services.

Another boost is for the Java programming language and of course for Sun Microsystems, the developers of Java. Ericssons gain in this project is that they deliver a SS7 stack, which the API uses, and being part of the project gives them a new technology to build future signaling stacks on.

The long-term goal from Ericssons side is to be able to provide a complete stack for the provider to purchase, including a simple to use API. This future stack should be so modular that when a new technology is developed they could easily exchange an old module for a new one.

2.2 Problems

Since the project is started by Sun Microsystems, the JAVA programming language is used to implement the API, CTS and the Reference Implementation (RI) parts of the system. We have not taken any Java courses at the University even though there have been such a course. Due to the interest in Object-Oriented Programming (OOP) we both have some knowledge that served us well during the construction and implementation of the CTS. Java is not that difficult to learn if you know some C++, the similarities become quite obvious very fast. The Java programming language is also quite easy to learn and it doesn't take the average C/C++ programmer long to become a very skilled Java programmer.

Another thing that one could think of as a problem is that Ericsson is a large corporation that has it's own standards and ways of doing things.

First we had to learn how write code in an Ericsson specific manner. To learn how to code for Ericsson we had to read through a stack of about 50 pages. Everyone that programs have their own little things they think make the code much easier to read. By having everyone writing the code in the same way, everyone can very easily read the code that someone else has written. One may think that everyone could code like they want, but no one would be able to read the code after six months, not even the programmer. Learning how to code in a way that Ericsson wanted took some time.

Storing the code was done using a program called ClearCase on a server somewhere. The storing of the source code in ClearCase was to maintain a certain consistency and to let all the people involved in the project to be able to edit the same code. This doesn't in itself cause a problem, but learning the complex structure of the program and the complexity of all catalogues and where everything should go was somewhat cumbersome.

2.3 Goals

The goal of the MAP API standardization is to provide the application programmer with a simple interface to the different functions of the already existing stack. In the future it should be possible to remove an old stack and replace it with a new without having to rewrite any of the code above or below, all this is made possible with OOP and Java.

Our job at Ericsson Infotech EIN/N is to implement a Conformance Test Suite (CTS) to test the MAP API standard on the client side in a predicable and controlled manner. As the name implies the CTS is constructed mainly to test the API, so the primary task is not to implement a fail-safe, commercial program but a program that can test all the details of the API standard.

We should also implement a demonstration version that can be used at showcases if we have the time. The demonstration application also includes a server part that both the CTS team the RI team has constructed. This server part is not a standard and is just implemented to have a demonstration model and a counter part for the client.

3 Prerequisites and demands

This section will present the prerequisites and the demands from Ericsson. We will explain the reasons for the prerequisites and how we fulfilled the demands.

We will also explain some how the prerequisites applied to the project.

3.1 Prerequisites

There were three prerequisites that Ericsson wanted us to fulfill.

The first prerequisite was that we had studied object orientated programming in Java or C++, which we had since we both took the course Advanced programming in C++. Since the project was to implement a conformance test tool for the Java API Integrated Network the use of Java, as a programming language should not be a surprise. The knowledge we already had sufficed far and it wasn't a problem that we hadn't programmed in Java before.

The second prerequisite was knowledge in English. This prerequisite was because we had to read a lot of manuals and other papers written in English, also all programs were in English. Most computer companies require English language knowledge since most terms in the computer world is in English.

To sum it up we didn't speak English to anyone due to the project but we had to read quite a lot of material in English. We had no problem with our English as far as we know.

The last prerequisite was a general knowledge in computer communication, which was fulfilled with the course Computer Communication I. A not so surprising prerequisite since the project deals with computer communication at a very large degree. It was mostly communication between clients and servers and the primitives sent between them as a means of communication.

3.2 Demands

The only demand that Ericsson Infotech had on our CTS application was that it should test each and every part of the SMS service on the client side of their MAP API. One must be able to generate test cases and then save them to disk with the CTS application. The possibility to load already saved test cases should also be implemented into the CTS application. Execution of test cases must also be implemented into the application. The ability to edit specific fields of the different primitives must also be implemented. The first CTS application for the TCAP API did not have the possibility to edit already added primitives and this proved to be quite cumbersome for the testers.

More of the prerequisites and demands are included in appendix Requirements Specification that is a translated version of what was presented to us before the start of the project.

4 Description of the construction solution

First of all we worked on a prototype for the GUI so we knew what was possible and not. When we were satisfied with the prototype we implemented the code that handles the method calls to the other classes in the API and RI.

The MAP CTS application was done as an evolutionary implementation where we ourselves tested the application and had others test the CTS application and present their ideas.

We used the JAIN TCAP CTS standardization project as a reference point and tried not to make the same mistakes as was done when developing the TCAP CTS application.

4.1 Communication Class diagram

The communication flows between the logical parts as shown in the diagram below. The CTS GUI, CTS Main and the SMSListener are part of our implementation on the CTS side of the project.

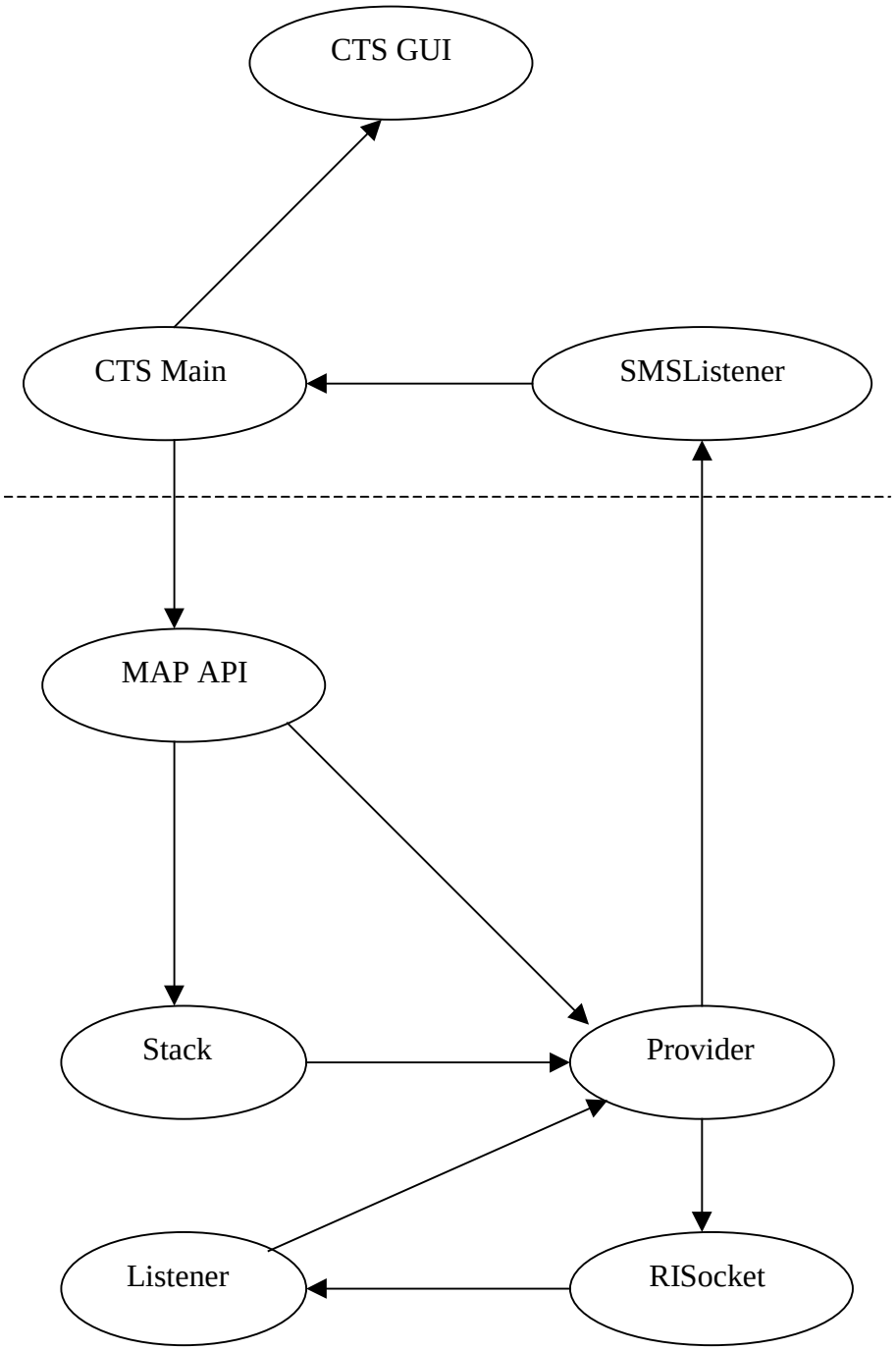

Figure 4.1: Communication Class Diagram

4.2 Thread Model

The following figure shows how the different threads run within the JAIN MAP API standardization project implementation.

The CTS implementation has one thread on either side and also a listener and validation part that is not a separate thread. The threads within the CTS is not instantiated explicit but running implicit from the Java Virtual Machine.

The RI and RMI threads are explicitly implemented as well as the queue for the RMI calls. The implementation and testing of the RI and RMI implementation is more described in the report [5].

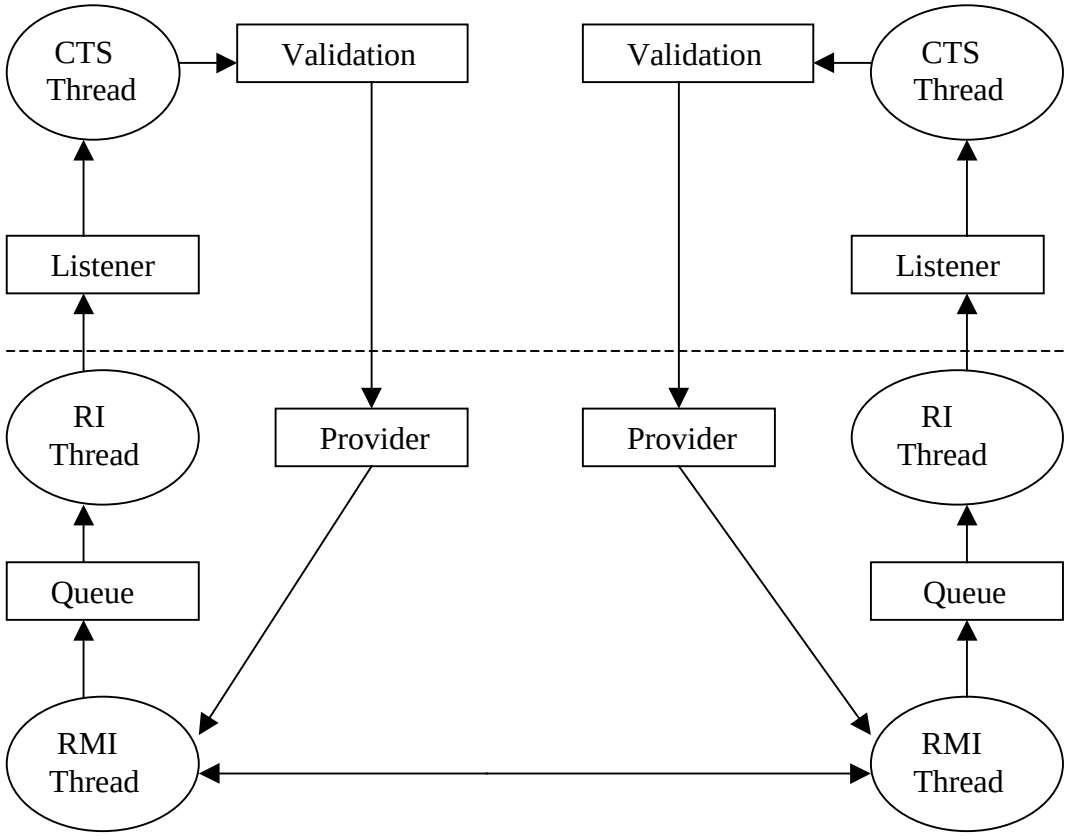

Figure 4.2: Thread Model

4.3 The application

The application is implemented to provide the user with a simple yet powerful interface to be able to test the JAIN MAP API standardization. A figure of the startup screen is shown below.

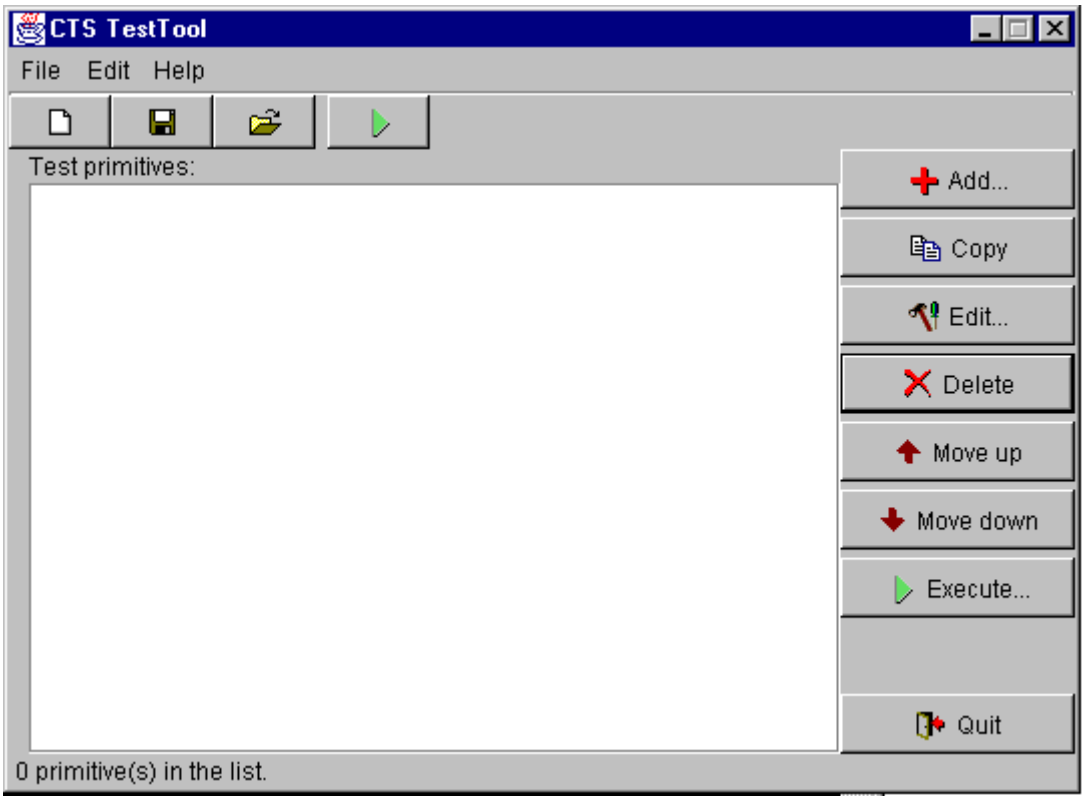

Figure 4.3: The application

Pressing any of the buttons where the name ends with three dots will bring up a new frame where other options can be selected. Some of the functionality is duplicated within the menus to satisfy users who like to use the menu options. The execute function is represented by the green triangle and duplicated twice as a button. The "white paper" button clears the list of primitives, while the disk saves and the folder opens a list of primitives.

Pressing the "Add" button will bring up the following frame where the users can select what to send to the server application.

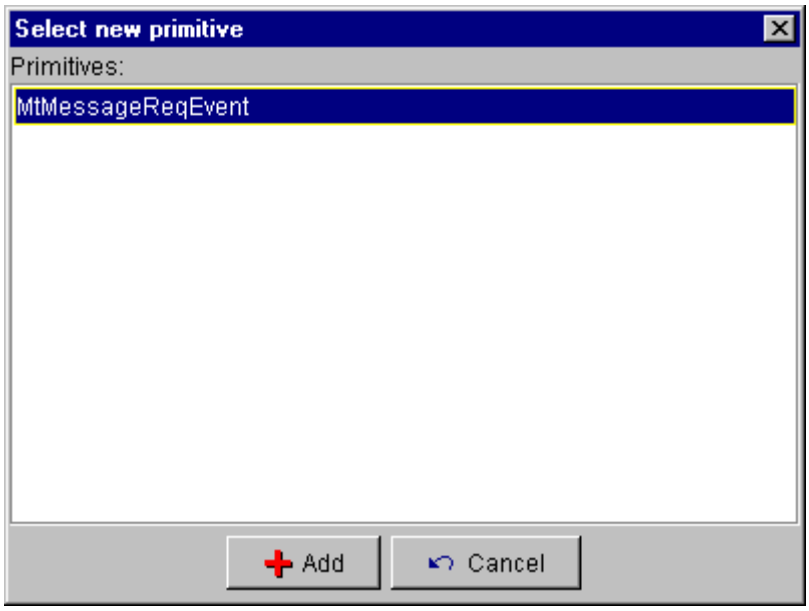

Figure 4.4: Adding a primitive

Pressing "Add" or double clicking on the primitive will add the selected primitive and pressing "Cancel" will cancel the add action and bring the user back to the start frame.

This simple interface is implemented in all parts of the application, mostly because there is no need to implement a more advanced interface than this.

5 Implementation and testing

Implementing and testing is one of the most interesting parts of developing the Conformance Test Tool. We will try to describe how we did the implementation and testing in this chapter.

5.1 Implementation

This section will describe the implementation of the different parts of the CTS. The ideas and thoughts of how to implement the parts will be described and what programs was used for what part.

5.1.1 Graphical User Interface

What we wanted was a program that looked and felt like any other Windows program. Another aspect was the depth of the frames, we chose to use a very shallow frame depth that made it easy to switch from one frame to another. The frame for editing a primitive's property was to be dynamic so if there was a change in the primitive classes there would be no need to recompile the CTS code. Building a dynamic GUI for the editing of a primitive proved to be to difficult to complete before the deadline and has to be implemented after the Bachelor's project.

The implementation of the Graphical User Interface (GUI) was created using Borland JBuilder 3 Foundation and NetBeans2.1. Additional coding was done using Visual SlickEdit 5.0.

5.1.2 Listener

This part of the CTS listens to incoming method calls from the server side and responds accordingly to them. The final version should allow the user to specify what the listener will respond when a certain primitive is sent.

The implementation was done using Visual SlickEdit 5.0 and compilation made easy by using Make in Java from the web site at the following adress:
http://www.ccs.neu.edu/home/ramsdell/make/edu/neu/ccs/jmk/index.html.

5.2 Testing

The testing of the program was done by having someone who was not involved in the coding, test the different functions of the program and then report to us what improvements could be done. Any bugs that the tester found was also fixed as soon as possible. As testers we used Kristofer Wiklund and Harald Grytberg who were developing the reference implementation for the SS7 stack, a part of the JAIN MAP API standardization project.

We did the initial testing during the development of the application. The first testing that we performed was mostly to make sure that the logic was correct and functioning. This testing is very important and difficult to perform, but has to be done to make sure that a functioning application is produced.

A request is sent from the CTS Client, containing all parameters that the tester/user sets in the request such as message and other parameters, to the client provider. The provider then calls the server listener on the machine/mobile phone where the server is running. The CTS Server then receives a indication from the listener. The server displays all the parameters to the user and then sends a response back to the Client through the server provider. The response can be defined by the tester/user at the server station. The Client receives a confirm and the parameters are displayed. This is how the API is tested. The tester/user can send more than one client request at a time, the server will then respond to all these indications with the same response. The RI generates a unique transaction id for all the primitives that are sent back and forth. These transaction ids are reused once the primitive has reached its destination and returned the id to the RI. All results from the test primitives can be saved in text files for later viewing.

The later testing that was performed by our test crew was to get feedback on the GUI.

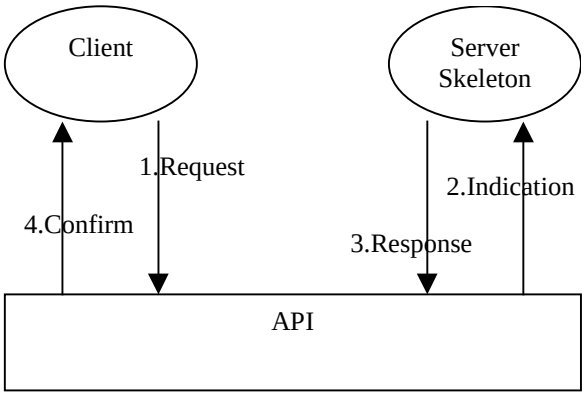

Figure 5.1: Primitives sent

6 Experience and recommendations

This section will cover topics that we think can be of use to others working on similar projects. We will give a brief comment on the programs that we have used, and how we tested our own application.

6.1 Programs

In this section we will give our evaluation of the different programs that we used during our project. Each program will be presented with its advantages, disadvantages and recommendations.

6.1.1 NetBeans 2.1 (free)

One thing that is good is that when you install NetBeans it searches for a already installed Java Virtual Machine (JVM) and uses the one it finds instead of installing it's own JVM.

After testing NetBeans 2.1 for a while we ran out of memory even though we had 128Mb of RAM. We started reading the help for NetBeans 2.1 and found out that NetBeans 2.1 was created with Java! A Java GUI Builder coded in Java (with a help system coded in Java)! Now we knew the reason for NetBeans being so slow and consuming so much memory.

NetBeans is a easy to use program but beware since it hogs memory and slows down your machine if your project starts getting to big. The program is quite easy to learn and comes with an extensive tutorial where one learns the basics and also learns how to make a bean. NetBeans 2.1 is a great program for developing graphic applications and you don't really know what you are doing, but if you want to tweak the code some, the program limits the user very much. Parts of the generated code is "locked" for the user so editing in generated sections is not possible within the NetBeans editor. You can bypass the "locked" sections if you use a different editor and you can even "unlock" the "locked" sections.

Another annoying thing is that the mouse wheel doesn't work to scroll through the code, which happens to be true for most of the Java based applications.

We recommend having a minimum of 128Mb of memory and a machine faster than a Pentium III 500MHz to be able to run this program at an acceptable speed.

6.1.2 Borland JBuilder 3 Foundation (free)

First thing you should do when installing Borland's JBuilder 3 Foundation is to remove any installed JVM since if it cannot install its own JVM, it will not work properly. To be completely sure that it will work you should remove JDK if you have it installed.

We used the free version of the program and it has some limitations, one thing is that you cannot make Java beans with it. It's easy to create a graphical user interface and you are able to edit the code within the program. The generated code is very messy and it takes a while before one fully understands it. The program worked well on a Pentium III 500MHz with 128Mb of memory.

6.1.3 Visual SlickEdit 5.0 (commercial)

This program doesn't require any Java Development Kit (JDK) to be installed and it doesn't install one for you either. This is more of a standard editor for pretty much all kinds of programming languages including Java. It's recommended that you install JDK with a JVM before installing Visual SlickEdit 5.0 because the install process will search your drives for an installed programming language and adapt the installation after that. We didn't use Visual SlickEdit 5.0 as much in the beginning as we used the other programs listed. Later on in the project when the more complicated parts were implemented we used only Visual Slickedit. It's an easy editor to learn that is very powerful and fast once you get the grasp of it, which doesn't take long. This editor should run very well on almost any PC, it's a small, fast and flexible editor.

6.1.4 ClearCase (commercial)

This program is used to keep track of program versions that are distributed on a server. It is a sort of a database program where the users get their own view on a disk. This means that if one of the project members delete some files, no one else that has those files on their view will notice this, only the one who deleted the files. When a user is done with some files and wants to share the files with others, the files are checked in so that everyone can use them. While a file is checked out only the user that checked out the file will benefit/suffer from changes done to the file by the user who checked out the file.

The program is very complex and very difficult to grasp for a beginner, but once you get the hang of it it's very powerful and versatile.

6.2 Testing

Testing your own program is not as easy as it may seem. Since you know exactly how it works it’s difficult to do something that you couldn’t think of doing. One aspect of this is that the coder will probably do the best white box testing.

Someone that has little knowledge of the program but no knowledge of the actual code does the best blackbox testing in our opinion.

When building a GUI it’s recommended having a few others test the layout and functionality since different users use the GUI in a different way. Some users prefer to use buttons while others use the menus.

7 Conclusions

Working on something as new as the JAIN MAP API standardization is both exciting and frustrating. The fun part is that it's brand new and you get possibility to put your personal touch into it and say your meaning in how and what it does. The frustrating part is that when you don't know what to do there is no one to ask either; you have to figure it out yourself.

Having the test tool implemented in Java is a bit of a bore since when you implement a method you have to use someone else's method. Using someone else's method to make you own work slows it down considerably. The reason to use Java could be that it will work on all platforms and it is a good candidate to be compatible with future programming languages.

When a company asks for some prerequisites it's because they need to have those prerequisites to do the job at hand. If you meet the requirements it's a good idea to review what you need to know just to freshen up your knowledge. Knowing your stuff gives you an edge and helps you solve the problems much easier.

Constructing an application is one of the most interesting and time-consuming parts of developing an application. This part of the process lets you decide what you think is important to have in the application and how it should work. Great care must be taken not to overdo it so that the time it would take to implement the ideas exceeds the time allocated.

After the construction part is done one must implement and test the program. Implementing shouldn't take much time if you did a good construction plan, the time spent developing the blueprints for the program is saved when you implement the plans. Beginners tend to start implementing without planning anything, this has the effect that the implementation takes twice as long as it needed to. The bottom line is that construction pay off when you start implementing.

Testing is done during the implementation part but also done when the application is completed. The post testing is done so that any eventual logic or programming errors can be caught before the shipment is done.

References

[1] Matti Drisin, Ericsson Infotech AB. JSR-000029 JAIN MAP Specification. http://java.sun.com/aboutJava/communityprocess/jsr/jsr_029_map.html August 11 1999

[2] JSR-000011 JAIN TCAP Specification http://java.sun.com/aboutJava/communityprocess/first/jsr011/index.html

And links from that index page

[3] Ericsson Infotech AB. Infotech Development Model.

[4] John D. Ramsdell, Make in Java. http://www.ccs.neu.edu/home/ramsdell/make/edu/neu/ccs/jmk/index.html

[5] Harald Grytberg and Kristofer Wiklund, JAIN MAP API RI. June 2000

Appendix

A Requirements Specification

A.1 Background

Sun runs a standardization work called JAIN (Java API:s for the Integrated Network) where they develop Java API:s for different "telecom protocols". An example of this telecom protocol is MAP (Mobile Application Part). Within the limits of JAIN Ericsson is leading a working team that develops a Java API for MAP. The functions of JAIN MAP API contain transmit/receive message like SMS and to get information about the mobile phone (ex. it's state and position). It's therefore a rather simple API with about five operations.

A.2 Task

In conjunction with the development of the API there should be a development of a test application that uses the API. This test application, which is called CTS (Conformance Test Suite), will generate test calls to the API and analyze the answers that return. The CTS consists of a MMI to control the tests and logic to generate test sequences.

The CTS shall be used to test that the JAIN MAP API from the stack provider follows the standard, while a RI (Reference Implementation) will be used by the ones that implements applications for the JAIN MAP API. One aspect is that the CTS can be tested against the RI and the other way around.

Our project is to develop the CTS and a parallel project is to develop the RI. The two projects will be closely related since one is testing against the other. The test configuration consists of one or more CTS' are run against one RI. If time allows there will be performance tests of both parts.

The two projects will be an integrated part of a larger project and will be run in parallel as the actual API is developed. The tests of the CTS and RI bring with it that the API in itself is tested, and from the tests the standard for the API can be changed.

There will also be a demo version for the JAIN MAP API that can be shown on expositions. A goal is also to develop the CTS and RI so that as much as possible can be used in the demo version. Another way is that if the project is completed before time there can be a development of a demo version within time.

A.3 To sum up the examination job parts

- In consultation with the staff at EIN do a system design for the CTS
- Write a specification in English for the CTS
- Implement the CTS
- Test the CTS against the RI including the API sources.
- Possibly a performance test (if time allows).
- Possibly develop a demo version (if time allows).

There is no security level on the project except possibly restrictions on the publication of source code.

A.4 Approximate time plan

January	Installation and studying
February	System design and specification
March	Implementation
April	Tests and modifications
May	Documentation (possibly development of demo version)

The most important prerequisites to be able to complete the project is:

- Object oriented programming in Java or C++

- Knowledge in English
- General knowledge in computer communication.

A.5 Purpose

The purpose of this examination project is to develop a good API-standard and test it. To do that we implement the CTS so it can test the API and RI.

B Abbreviations list

API - Application Programmers Interface

CTS - Conformance Test Suite

GUI - Graphical User Interface

HLR - Home Location Register

JAIN - Java API Integrated Network or Java Advanced Intelligent Network

JDK - Java Development Kit

JVM - Java Virtual Machine

MAP - Mobile Application Part

MMI - Man Machine Interface

OOP - Object Orientated Programming

PLMN - Public Land Mobile Network

RI - Reference Implementation

RMI – Remote Method Invokation ???

SMS - Short Messages Service

SS7 - Signalling System 7 ????

TCAP - Transaction Capability Application Part