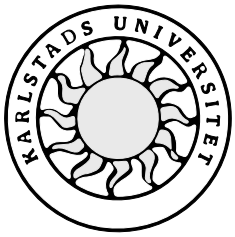


Datavetenskap

Catharina Eng

Cecilia Stark

Messaging API till managementfunktionen för SS7

Examensarbete, C-nivå

2001:04

Messaging API till managementfunktionen för SS7

Catharina Eng

Cecilia Stark

Denna rapport är skriven som en del av det arbete som krävs för att erhålla en kandidatexamen i datavetenskap. Allt material i denna rapport, vilket inte är mitt eget, har blivit tydligt identifierat och inget material är inkluderat som tidigare använts för erhållande av annan examen.

Catharina Eng

Cecilia Stark

Godkänd, 5 juni, 2001

Handledare: Johan Garcia

Examinator: Stefan Lindskog

Sammanfattning

Detta dokument beskriver hur felmeddelanden som uppstår i en SS7-stack kan skickas via SMS till mobiltelefon. Rapporten förklarar bakgrunden och syftet med uppgiften. Det ges vidare en allmän introduktion om telekommunikationsnätverk, GSM-nät, signalering och SS7. Rapporten beskriver också de viktigaste delarna i en SS7-stack. Vidare förklaras olika lösningsalternativ, för- och nackdelar med dessa och den valda lösningen. Det ges även en kort information om programmeringsspråket Java, eftersom den valda lösningen implementerades i detta. Konstruktionslösningen och omgivningsmiljön beskrivs genom klassdiagram och sekvensdiagram. Utifrån dessa konstruerades sedan applikationen. Den utvecklingsmiljö applikationen använder sig av presenteras. För att förklara strukturen på applikationen, det vill säga vad applikationen innehåller för klasser, förklaras även begreppen inre klasser och metoder. En kort beskrivning av hur testning går till, ges. Slutligen analyseras och summeras arbetet.

Messaging API to the management function for SS7

Abstract

This document describes how error messages occurring in a SS7-stack can be sent through SMS to a mobile phone. The report explains the background and purpose of the assignment. Further it gives a general introduction to telecommunication networks, GSM networks, signaling and SS7. The report also describes the most important parts in a SS7-stack. Further it explains the different solutions, their advantages and disadvantages, and the chosen solution. Since the chosen solution is implemented in the programming language Java, the report will give some information about the language. The design and environment are described by class diagrams and sequence diagrams. From these diagrams the application was constructed. The development environment of the application is introduced. To explain the structure of the application, that is what classes the application contains, inner classes and methods are described. A short description of the testing is also given. Finally the work is analyzed and summarized.

Tack

Vi vill rikta ett stort tack till Sören Torstensson och Johan Garcia som har varit våra handledare under detta examensarbete. Vi vill även tacka Birgitta Sjöqvist Eriksson, Ulf Hellsten och Jens Landgré som har varit oss behjälpliga i vårt arbete.

Innehållsförteckning

1 Inledning 1

2 Bakgrund 2

2.1 Problem / uppgift 2

3 Telekommunikation - introduktion 3

3.1 Telekommunikationsnätverk 3
3.1.1 GSM

3.2 Signalering 5

3.3 SS7 6

3.4 SS7-stacken 7
3.4.1 MTP
3.4.2 SCCP
3.4.3 TUP
3.4.4 TCAP
3.4.5 ISUP / INAP
3.4.6 MAP

3.5 SMS 10

3.6 Portable SS7 10

4 Lösningsalternativ 10

4.1 Java 11

4.2 Uppgiften 11

4.3 Den valda lösningen 12

5 Beskrivning av konstruktionslösning 12

5.1 Design 14
5.1.1 ROC API
5.1.2 Java MAP API

6 Implementation 18

6.1 Utvecklingsmiljö 18

6.2 Applikationens struktur 18
6.2.1 Inre klassen *AlarmListener*
6.2.2 Metoder

7 Test **20**

8 Analys **20**

8.1 Arbetets framskridande 20

8.2 Framtiden 20

9 Slutsats **21**

Referenser **22**

A Ordlista **23**

Figurförteckning

3.1 User- och controlplane 4
3.2 Cellformig struktur 5
3.3 SS7-stacken i jämförelse med OSI-modellen 6
3.4 Exempel på SS7-stack 7
3.5 Telefonnätets olika delar 9
5.1 Applikationen och dess omgivningsmiljö 13
5.2 Alarmets gång 14
5.3 Klassdiagram ROC 15
5.4 Sekvensdiagram ROC 16
5.5 Klassdiagram Java MAP API 17
5.6 Sekvensdiagram Java MAP API 17

1 Inledning

Vi studerar vid Karlstads universitet på programmet för dataingenjörer. För att kunna få ut vår Högskoleingenjörsexamen ingår ett examensarbete på 10 poäng. Examensarbetet syftar till att återspegla de kunskaper vi har fått under 2 ½ år och att ge oss inblick i vad en dataingenjör kan tänkas arbeta med.

Ericsson Infotech, avdelning N, Karlstad, hade ett uppslag till examensarbete som vi ansåg vara intressant. Företaget vill ha hjälp med att utreda och implementera en felrapporteringsapplikation. Om resultatet visar sig falla väl ut, avser företaget eventuellt att använda sig av denna applikation.

För att på bästa sätt kunna tillgodogöra sig denna rapport är det fördelaktigt med grundläggande kunskaper inom datavetenskap och datakommunikation.

I kapitel 2 presenteras uppgiften och dess bakgrund. Kapitel 3 tar upp området telekommunikation. Här beskrivs hur ett telekommunikationsnät är uppbyggt och hur kombinationen av signaleringspunkter och dess sammankopplande signaleringslänkar bildar ett signaleringsnätverk. Vidare förklaras GSM och signalering kortfattat. I detta kapitel beskrivs även SS7-protokollet. Vilket är en internationell standard bestående av en samling protokoll som bildar en stack. Kapitlet avslutas med en kort beskrivning av SMS och den portabla SS7-stack som Ericsson Infotech erbjuder. I kapitel 4 beskrivs de olika lösningsalternativen och deras för- och nackdelar. Vidare förklaras här ROC, Java och den valda lösningen. Beskrivning av konstruktionslösning och design finner man i kapitel 5. Applikationens utvecklingsmiljö och struktur beskrivs i kapitel 6. Kapitel 7 beskriver kortfattat hur applikationen kan testas. Analys av arbetet, arbetets framskridande och framtid presenteras i kapitel 8. Slutligen summeras arbetet i kapitel 9.

2 Bakgrund

I dagens samhälle används telekommunikation i stor utsträckning. Därför måste tillförlitlighetskraven i telekomutrustning vara extremt höga. Detta medför att kraven på att övervaka all utrustning i nätet är stora och att man snabbt får larm om felsituationer. Bl.a. måste signaleringen övervakas. Med signalering menar vi de signaler som sänds mellan de olika noderna i nätet. Dessa signaler används bland annat för att koppla upp och ner förbindelser.

Signaleringsprotokollen i dagens nät är baserade på en internationell standard. Denna standard heter SS7 (Signalling System No.7).
Ericsson Infotech är ett av många företag som utvecklar protokoll enligt SS7-standarden och de erbjuder ett antal produkter som är baserade på SS7.

2.1 Problem / uppgift

Ericsson Infotech hade önskemål om en applikation som ska skicka meddelanden om de eventuella fel som uppstår i den SS7-stack, utvecklad av Ericsson. Rapporteringen kan till exempel ske genom att ett SMS-meddelande (Short Message Service) skickas till support.

Uppgiften är att göra en applikation som ska ta hand om de felmeddelande som uppstår i systemet och skicka dessa via SMS och email till support. Det bestämdes att den del som skulle utföras först var SMS-funktionen och om tid fanns så skulle även email-funktionen implementeras.

3 Telekommunikation - introduktion

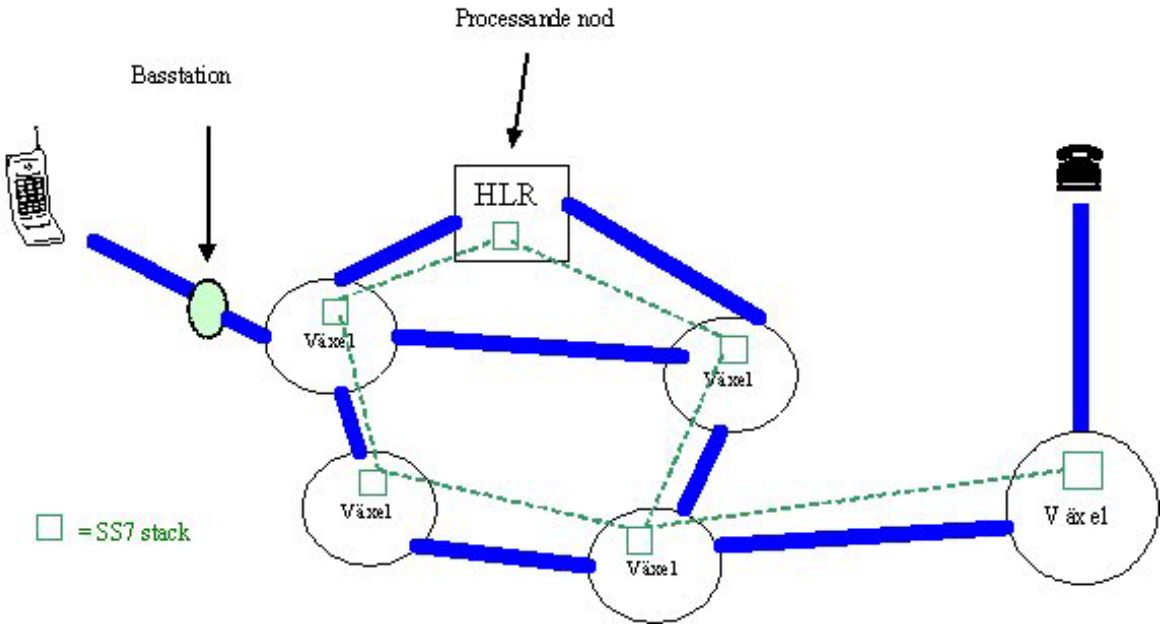
”Telekommunikation är kommunikation som sker på avstånd på annat sätt än genom att fysiska informationsbärare (till exempel papper eller disketter) transporteras från avsändare till mottagare. Vid telekommunikation överförs informationen, som kan bestå av text, ljud, stillbilder eller rörliga bilder och kombinationer därav, med hjälp av elektromagnetiska vågor [10].” Det går att dela in de olika telenäten. Man skiljer mellan ledningsbundna nät och sådana nät som använder sig av etern, allmänt tillgängliga nät och interna nät samt fysiska nät och logiska nät. Fysiska nät utgörs av kablar, radiolänkar, satelliter med mera medan logiska nät utgörs av den särskilda utrustning som krävs för en viss tjänst, till exempel telefoni.

3.1 Telekommunikationsnätverk

Ett telekommunikationsnätverk är uppbyggt av ett antal *switchande* noder och ett antal *processande* noder, vilka är sammankopplade via överföringslänkar. För att upprätta en förbindelse mellan de *switchande* noderna enligt SS7 protokollet, krävs vissa funktioner i noderna. För att förklara hur ett SS7-nätverk fungerar kan man se det som två olika plan, ett user- och ett control-plane. Controlplane har hand om upp- och nedkoppling av förbindelserna. Controlplane utgörs av SS7-noder (signaleringspunkter) och signaleringslänkar mellan noderna. Här skickas även SMS-meddelandena. Userplane är där användarna sänder sin data, exempelvis så är det på detta plan som rösten färdas vid telefonsamtal.

Kombinationen av signaleringspunkter och deras sammankopplande signaleringslänkar bildar ett SS7 signaleringsnätverk. SS7 har applikationsprotokoll som sköter signaleringen för röst, mobil data och intelligenta nätverk.

Figur 3.1 nedan visar en förenklad version av ett SS7 nätverk.

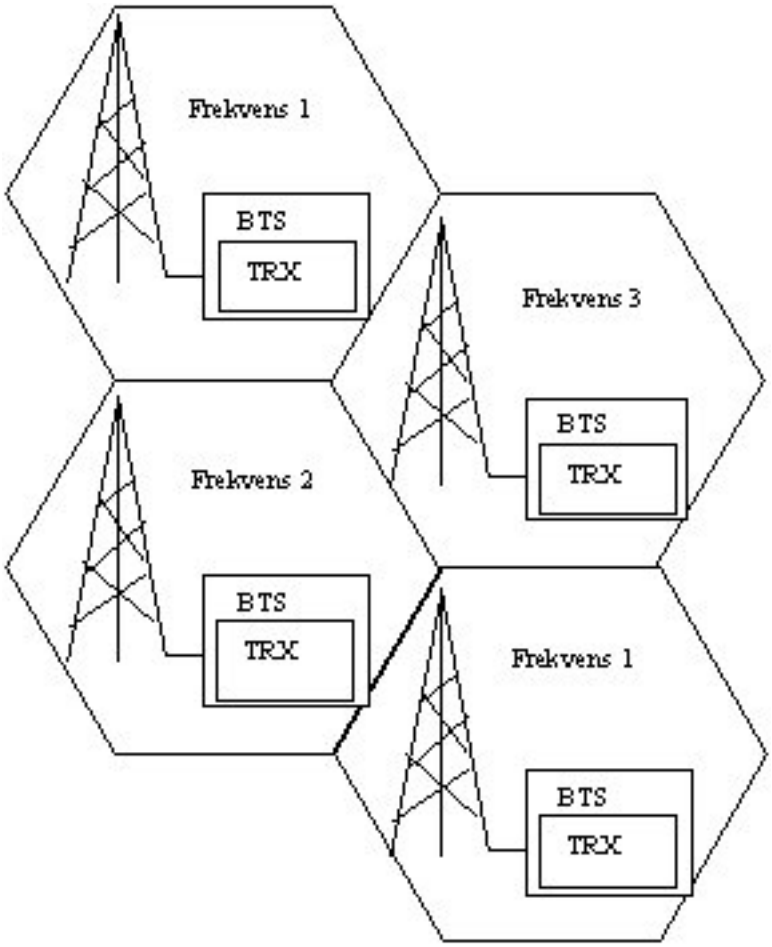


Figur 3.1. Den tjockare linjen symboliserar den väg t.ex. rösten färdas och kan benämnas User Plane. Den streckade linjen symboliserar den väg signaleringsmeddelandena färdas mellan SS7 stackarna och kan benämnas Control Plane.

3.1.1 GSM

GSM (Global System for Mobile Communication) är världens mest använda standard för mobila digitala nät. Det är en öppen standard som utvecklats under överinseende av telekommunikationsbranschens internationella standardiseringsorgan.

Liksom alla moderna mobila telefonnätverk använder sig GSM av en cellformig struktur, se figur 3.2.



Figur 3.2. Cellformig struktur

Grundidén med ett cellformat nätverk är att dela upp det tillgängliga frekvensområdet, och bara tilldela vissa delar av frekvensspektrumet till en cell. Eftersom varje cell bara har en basstation (BTS) med begränsad räckvidd kan man återanvända frekvenserna.

3.2 Signalering

Signalering är det telekommunikationsspråk (protokoll) som maskiner och datorer använder för att kommunicera med varandra. Signalerna som en användare matar in (till exempel telefonnummer) måste konverteras till ett format som passar maskinerna och sedan överföras. Signalering överför ”styrinformation” men normalt inte någon användardata (payload). Ett undantag är dock SMS där användardata (små textmeddelanden) överförs i signaleringsnätet. Signalering kan jämföras med piloterna och flygpassagerarna på ett flygplan. Besättningen är inga ”payload”, men de är nödvändiga för att styra ”the payload”. [2]

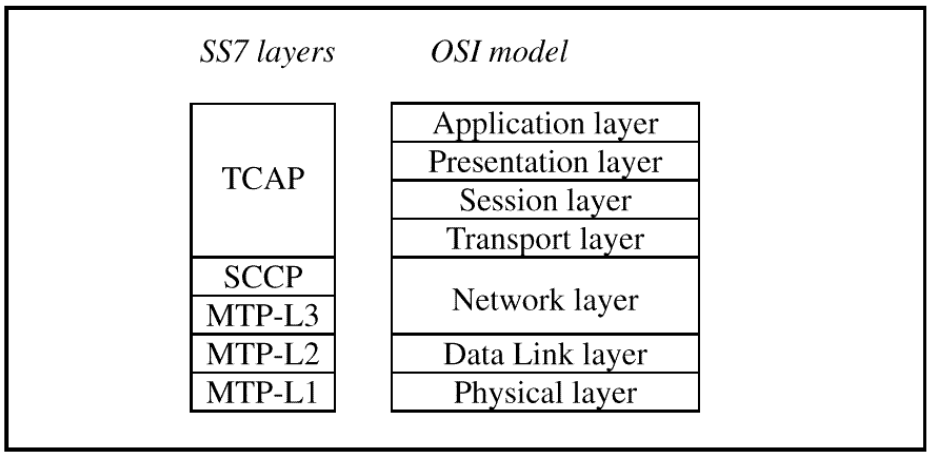
I ett SS7-nätverk finns det olika noder till exempel SSP (Service Switching Point), SDP (Service Data Point) och SCP (Service Control Point) mellan vilka signaler skickas.

3.3 SS7

Dagens kommunikationsnät baseras på en internationell standard, SS7. Denna standard består av en samling protokoll som tillsammans bildar en stack. En SS7-stack kan bland annat innehålla följand protokoll:

- MTP - Message Transfer Part
- SCCP - Signaling Connection Control Part
- TCAP - Transaction Capabilities Application Part

SS7 protokollet motsvarar fys-, länk-, nätverks-, och applikationslagren i OSI-modellen, se figur 3.3.



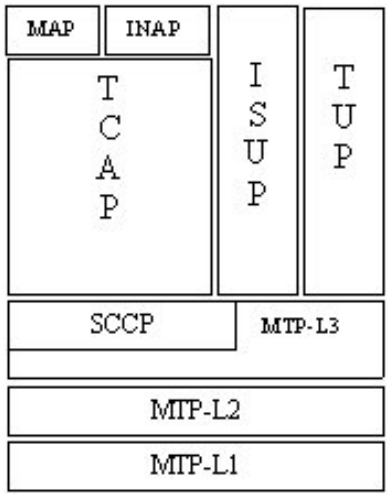
Figur 3.3. Visar hur SS7-stacken förhåller sig i jämförelse med OSI-modellen.

Från början var SS7 enbart avsett för att ersätta den gamla tekniken för telefonsamtal men detta har utökats betydligt och SS7 används numera inom andra områden som tex. ISDN (Integrated Services Digital Network) för både röst och data signalering.

3.4 SS7-stacken

En SS7-stack är uppbyggd av olika protokollager. Varje lager har sin funktionalitet. Att notera är att man i en SS7-stack kan utesluta vissa delar av ett lager. På översta lagret kan man exempelvis använda sig endast av TCAP.

Figur 3.4 nedan visar ett exempel på hur en SS7-stack kan vara uppbyggd.



Figur 3.4. Exempel på SS7-stack.

3.4.1 MTP

MTP (Message Transfer Part) är det lager som handhar meddelandeöverföringen. Bilden ovan visar tre olika MTP-lager; 1, 2 och 3.

MTP lager 1 kan man direkt härleda till OSI-modellens fysiska lager. MTP lager 1 definierar den fysiska, elektriska och funktionella karaktäristiken på den digitala länken. Lagret ansvarar även för överföring av binära bitar från en signaleringspunkt till en annan.

Lager 2 motsvarar OSI-modellens länklager. Det ansvarar för point-to-point överföring av ett meddelande över en signaleringslänk. Här implementeras även flödeskontroll, meddelandekontroll och felkontroll. MTP lager 2 har kontroll på närmaste noden, men känner ej till övriga noder.

Det tredje och sista MTP-lagret motsvarar nätverkslagret i OSI-modellen. Detta lager kan ”se” nätverket och dess noder och vet vilka funktioner dessa noder har (inom rimligt avstånd). Här tillhandahålls även meddelanderouting mellan signaleringspunkterna i nätverket, det vill säga den omdirigerar trafiken för att undvika skadade länkar och signaleringspunkter.

3.4.2 SCCP

SCCP (Signaling Connection Control Part) lägger till ytterligare adresseringsmöjligheter utöver det som MTP lager 3 erbjuder. Lagret tillåter användandet av Subsystem Numbers (SSN) för att adressera specifika applikationer. Detta lager kan bestämma meddelandets destination genom att använda globala titlar (Global Titles - GT). SCCP tillhandahåller förbindelselös och förbindelseorienterad nätverksservice.

3.4.3 TUP

TUP står för Telephone User Part och är ett gammalt protokoll för de vanligaste telefontjänsterna. Här sätts en förbindelse upp, den underhålls och kopplas ner. Det finns många olika, nationella versioner.

3.4.4 TCAP

TCAP (Transaction Capabilities Application Part) tillhandahåller support till applikationer som behöver utväxla information som inte är relaterad till ett telefonsamtal. Exempel på sådan information är databasförfrågningar och sändande av SMS meddelanden. Användare av TCAP kan göra transaktioner mot noder i annan operatörs nät. TCAP stödjer strukturerade och ostrukturerade dialoger och använder sig av SCCPs förbindelselösa service.

3.4.5 ISUP / INAP

ISUP (ISDN User Part) ersätter TUP. De flesta funktionaliteter som finns i TUP finns även med i ISUP. ISUP erbjuder fler tjänster än TUP, bland annat konferenssamtal, samtal väntar, nummerpresentation med mera.

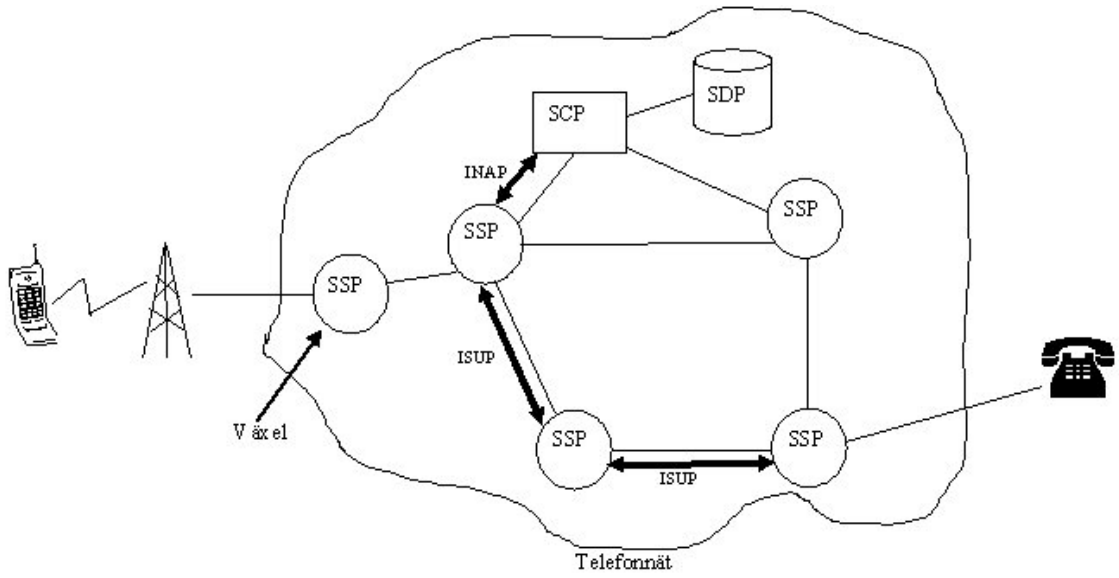
INAP (Intelligent Network Application Part) medger snabb och centraliserad implementation av olika sorters service såsom:

- Gratistelefoni, kreditkortssamtal etc.
- Virtuella privata nätverk
- Personliga nummer
- Telefonröstning
- Telefonväckning

INAP introducerar två Intelligent Network (IN) SS7 noder:

- SCP *Service Control Point.* Denna nod tillhandahåller tjänstelogik (program) och eventuellt data från databaser, så kallade SDP (Service Data Point) .
- SSP *Service Switching Point.* En SSP skickar signaleringsmeddelanden till andra SSP-noder, för att sätta upp en förbindelse, underhålla denna och koppla ner den.

Figur 3.5 nedan visar telefonnätets olika komponenter. Om man till exempel ringer ett 020-nummer, kommer signalen först till en växel (SSP). SSP vet inte vad ett 020-nummer är, eftersom detta inte är ett ”riktigt” telefonnummer. Via protokollet INAP skickas därför signalen till en SCP. SCP frågar sedan SDP, som är en databas, vilket det riktiga telefonnumret är. Detta nummer (till exempel 054-193441) skickas sedan till SSP som därefter kopplar upp ett samtal via ISUP-protokollet.



Figur 3.5. Telefonnätets olika delar

3.4.6 MAP

MAP (Mobile Application Part) definierar de signaleringsfunktioner som används i GSM mobila nätverk. Tillhandahåller bland annat följande service till den som använder MAP:

- Mobil service, det vill säga service för att hålla reda på vart mobiltelefonen befinner sig med mera.
- SMS

3.5 SMS

SMS (Short Message Service) är en standard för att sända och ta emot textmeddelanden i mobiltelefoner. SMS sänds som små textpaket som kan innehålla 160 tecken och som klarar de flesta språk. Grundfunktionen är att sända ett textmeddelande från en mobiltelefon till en annan. När det sänds ett SMS meddelas GSM-nätet om att ”Nu vill jag sända ett SMS”. Därefter kontrollerar nätet att allt är okej för koppling. Nätet tar SMS-textpaketet och vidarebefordrar det till den angivna adressen [7]. SMS-meddelanden kommer fram till en telefon helt utan att telefonens ägare behöver lägga sig i. Det är nätet som bestämmer om det är klart att sända meddelandet eller inte [8].

3.6 Portable SS7

Ericsson Infotech erbjuder en *Portable SS7* stack vilken är konfigurerad för olika systemarkitekturer. Modulerna i en Ericsson SS7-stackprodukt är programmerade i ANSI C (ANSI /ISD9899-1990). Den modulära strukturen på en portabel SS7 stack tillåter flera variationer för att möta de specifika behov som finns. [5]

4 Lösningsalternativ

Den givna uppgiften (applikationen) kunde implementeras på två olika sätt: med hjälp av C eller Java. För att kunna lösa uppgiften måste två olika gränssnitt användas, ett mot ROC (Reusable OAM Component) och ett mot MAP. Ericsson Infotech använder sig av flera olika management API:er (gränssnitt).

En ROC består av ett uppsättning JavaBeans som tillsammans bildar en trädstruktur. JavaBeans är den komponentteknik som finns för Java, och innebär att en kodmodul enkelt kan paketeras, konfigureras, distribueras och sedan användas i en mängd olika applikationer. ROC:en används för att konfigurera och underhålla ett lager (t.ex. MTP-L3, SCCP, TCAP...) i en SS7 stack. Den kan även ha stöd för alarm, order och statistik. ROC:arna laddas dynamiskt in i DCM och kan där kombineras ihop till en komplett stack konfiguration. DCM är ett grafiskt konfigurationsverktyg och det används för att konfigurera och distribuera SS7-stacken.

MAP API:et handhar kommunikationen i en stack. Båda API:erna inspireras av Javas events-hantering.

4.1 Java

Java är ett plattformsoberoende och objektorienterat programmeringsspråk. Med plattformsoberoende menas både oberoende av en viss processor, ett visst operativsystem och ett visst användargränssnitt. Java är fullt ut objektorienterat i och med att det har stöd för klasser och objekt, arv och polymorfism.

Java har en filosofi att göra hårda kontroller vid kompileringen så att fel upptäcks redan då och inte under exekvering av ett program. Genom att pekare inte finns med i språket undviks många av de typproblem som är vanliga i andra språk. Programmeringsspråket Java underlättar skapandet av robusta program, det vill säga program som är förutsägbara och som inte kraschar. Många operativsystem har numera stöd för parallell exekvering genom så kallade trådar där ett program kan innehålla många samtidiga exekveringsvägar. Normalt är dessa mekanismer svåra att programmera, men detta underlättas om Java används. I språket finns inbyggt stöd både för att skapa och styra trådar, samt mekanismer för att synkronisera trådar med varandra.
I Java använder man sig bland annat av event-hantering. Med event-hantering avses att hantera de händelser som inträffar när ett program med användargränssnitt exekverar. [3] [4]

4.2 Uppgiften

Uppdragsgivaren hade önskemål om att applikationen skulle kunna skicka meddelanden via SMS och/eller email. Följande för- och nackdelar listades upp för alternativen:

SMS

+ Noderna i telefonnätet behöver ej vara anslutna till Internet.
- Om SS7-stacken går ner går det ej att skicka SMS.

Email

+ Förbindelsen förblir intakt om SS7-stacken går ner, vilket innebär en större säkerhet att meddelandet kommer fram.
- Noderna måste vara anslutna till Internet.

I den SS7-programvara som Ericsson Infotech utvecklar så kan olika felmeddelanden genereras. Felmeddelandena delas in i tre olika nivåer.

1. notify När detta tillstånd inträffar får man ett meddelande att något har hänt, dock inget allvarligt. Man kan till exempel säga att man vill ha ett *notify* efter ett visst antal skickade paket.

2. *warning* När tillståndet *warning* inträffar så skickas ett meddelande att något har gått fel, men systemet fungerar fortfarande. Om detta fel ej åtgärdas kan systemet krascha.

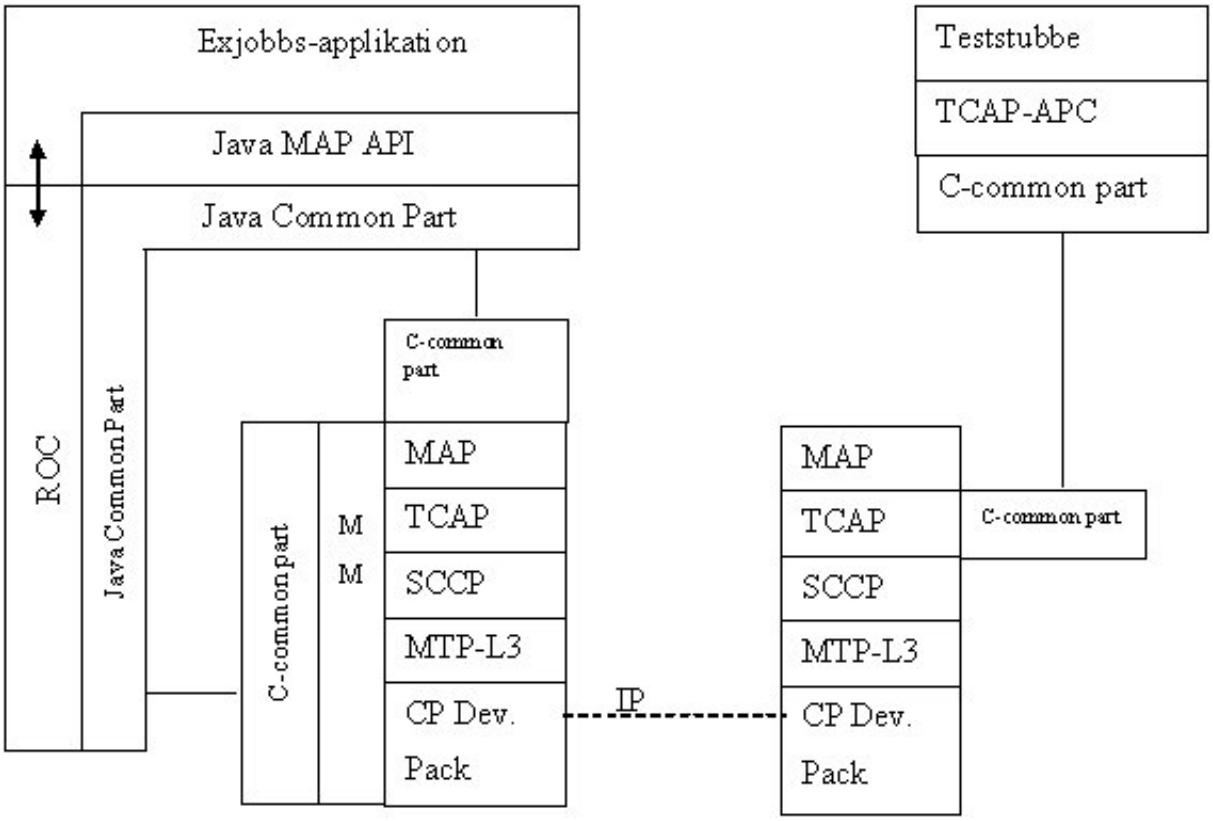
3. *fatal* Detta tillstånd indikerar att ett allvarligt fel har uppstått. Man kan ej förlita sig på systemets funktionalitet.

4.3 Den valda lösningen

Det programmeringsspråk som lämpade sig bäst för uppgiften var Java.
De management API:er som valdes var ROC API och MAP Java API. ROC API valdes eftersom det är det mest moderna API:et och är skrivet i programmeringsspråket Java. MAP Java API valdes eftersom Ericsson Infotech använder sig av detta gränssnitt mot SS7-stacken.
Det bestämdes att uppgiften skulle begränsas till att bara omfatta SMS. En följd av denna begränsning blev att endast *warning*- och *notify*-felmeddelanden kan användas på grund av att det endast är dessa som kan skickas via SMS. Via email skulle man dock kunna vidarebefordra alla tre typerna av felmeddelanden.

5 Beskrivning av konstruktionslösning

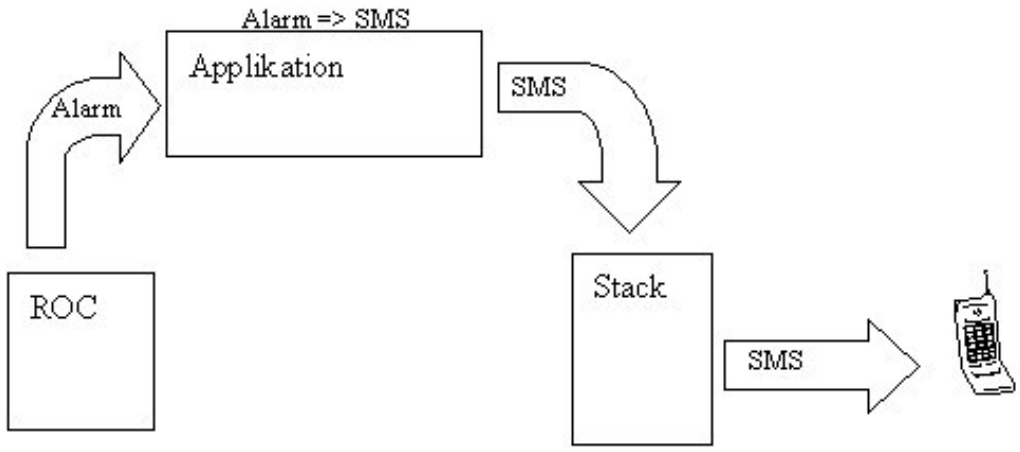
För att komma fram till en konstruktionslösning konstruerades en övergripande bild av applikationen och dess omgivningsmiljö. Denna bild visar hur vår applikation skall implementeras mot den befintliga stacken.
För att applikationen skall kunna implementeras och testas byggdes två olika stackar upp. En stack som vår applikation skall använda sig av och en på vilken en teststubbe implementeras. Dessa stackar förklaras ej närmare, eftersom detta inte är relevant för det fortsatta arbetet.
Figuren nedan visar applikationen och dess omgivningsmiljö.



Figur 5.1. Bilden visar en övergripande vy över applikationen och dess omgivningsmiljö.

Applikationen kommunicerar med både ROC och Java MAP API. I ROC initieras en lyssnare som hela tiden lyssnar efter nya alarm. Alarmet skickas till applikationen varifrån det skickas som ett SMS-meddelande via stacken till mobiltelefon. Detta sker utan inverkan från utomstående.

Figur 5.2 nedan visar alarmets gång från ROC till mobiltelefon.



Figur 5.2. Alarmets gång.

Utifrån figur 5.1 arbetades en design fram. Designen var en nödvändig del av utvecklingsarbetet för att få en förståelse för hur den nuvarande situationen såg ut. Dessutom kommer denna att kunna utnyttjas när implementationen av lösningen ska ske.

5.1 Design

Eftersom applikationen skall arbeta mot två olika gränssnitt, ROC API och Java MAP API, gjordes ett klassdiagram och ett sekvensdiagram mot varje gränssnitt.

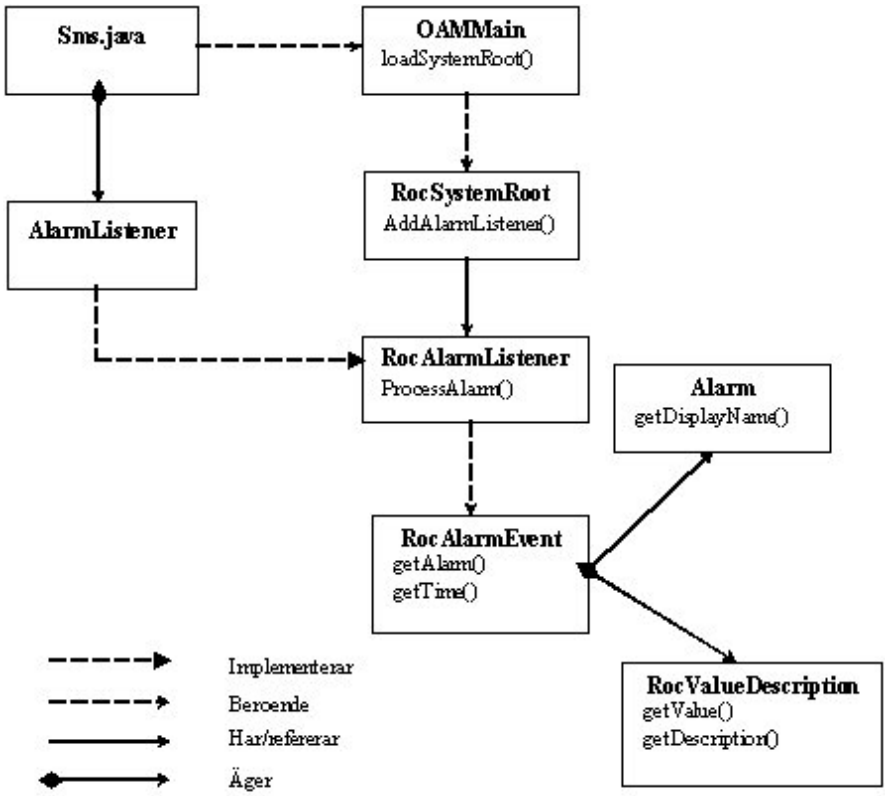
5.1.1 ROC API

Vid designandet av gränssnittet mellan applikationen och ROC API användes dokumentet ROC API [1]. Detta dokument innehåller designspecifikationen för SS7 OAM (Operation Administration and Maintenance) API, dvs. det gränssnitt mot vilket man programmerar applikationerna. I dokumentet visas med hjälp av bl.a. klassdiagram och sekvensdiagram hur de olika klasserna är relaterade till varandra. Utifrån detta klassdiagram skapades ett nytt, mindre klassdiagram med relevanta delar. Ur detta nya klassdiagram konstruerades ett sekvensdiagram.

5.1.1.1 Klassdiagram

Ett klassdiagram utarbetades för att visa hur de olika klasserna relaterar till varandra. Diagrammet visar hur applikationen relaterar till de berörda klasserna.

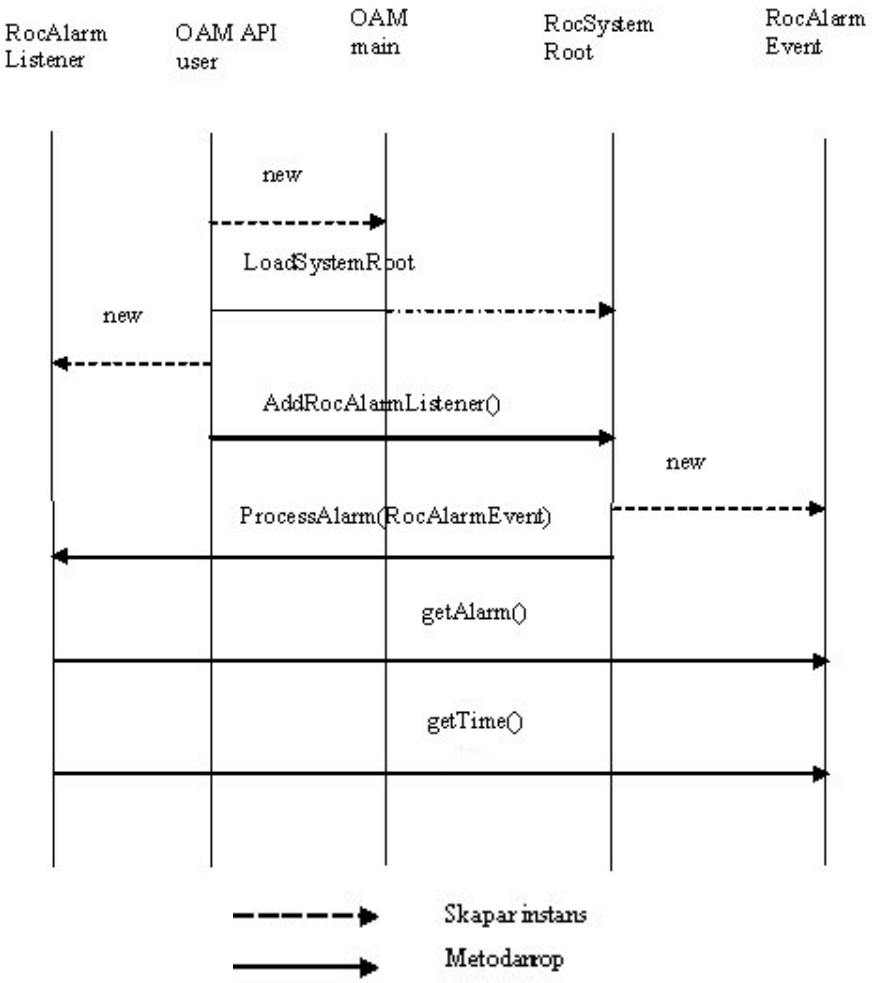
Figuren nedan visar klassdiagrammet.



Figur 5.3. Klassdiagram ROC

5.1.1.2 Sekvensdiagram

Sekvensdiagrammet gjordes för att få en mer detaljerad bild över hur anrop sker. Figur 5.4 visar vårt diagram. De vertikala linjerna visar de olika klasserna och de horisontella linjerna visar hur funktionsanropen sker.



Figur 5.4. Sekvensdiagram ROC

5.1.2 Java MAP API

Vid designandet av gränssnittet mellan applikationen och stacken konstruerades ett klassdiagram och ett sekvensdiagram. Vid designandet av sekvensdiagrammet använde vi oss av MAP ETSI R4 ETSI [9].

5.1.2.1 Klassdiagram

För att få en överblick över hur de olika klasserna, som används i applikationen mot Java MAP API, relaterar till varandra, konstruerades ett klassdiagram. Detta diagram ses i figur 5.5.

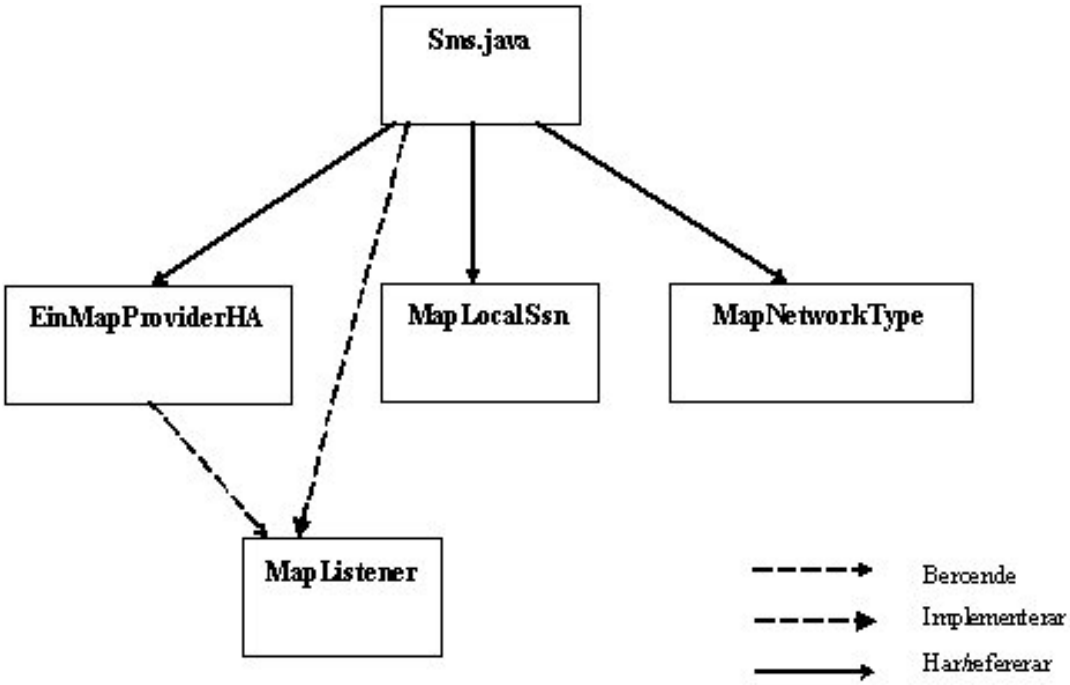
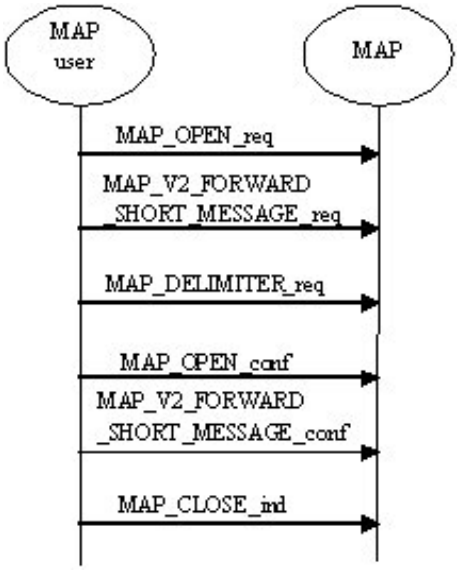


Fig. 5.5. Klassdiagram Java MAP API

5.1.2.2 Sekvensdiagram

För att få en mer detaljerad bild över kommunikationen mellan applikationen och stacken användes dokumentet MAP ETSI R4 Functional Specification API [6]. Utifrån uppgifter i detta dokument konstruerades sekvensdiagrammet nedan.



Figur 5.6. Sekvensdiagram Java MAP API

6 Implementation

Utifrån klass- och sekvensdiagram konstruerades applikationen. Applikationen består av en enda klass, en inre klass och ett antal metoder. Den utvecklingsmiljö som applikationen och dokumentationen utvecklades i är den som Ericsson Infotech tillhandahåller.

6.1 Utvecklingsmiljö

Applikationen och dokumentationen har utvecklats i följande utvecklingsmiljö:

- Sun SPARCstation 5 dator med 256 megabyte RAM och 2,1 gigabyte hårddisk.
- Solaris CDE version 1.2
- Open Windows
- Office Word 97
- X-emacs 21.4
- Paintshop PRO 7.01
- Netscape Communicator 4.72
- JDK (Java Development Kit) 1.3 for Solaris

För att kunna kontrollera om applikationen innehåller några buggar (fel) kan en debugger användas. I programspråket Java finns en debugger (JDB) som medföljer JDK. I detta projekt valdes dock att ej använda någon debugger eftersom applikationen inte är så stor. En annan anledning till detta beslut är att det fattades tid att införskaffa den kunskap som krävdes för att utnyttja programmet.

6.2 Applikationens struktur

Applikationen består av en klass, *Sms.java*. Denna klass består av en inre klass och flera metoder. En inre klass initieras när det finns behov av att definiera en klass internt i en annan klass. Dennan inre klass kan endast instansieras och användas inom den klass i vilken den är definierad. En inre klass kan välja att implementera gränssnitt eller ärva andra klasser, precis som vilken klass som helst. En metod i programmeringsspråket Java kan liknas vid en funktion i C/C++.

Klassen *Sms.java* använder sig av ett interface *MapListener*. Interface (gränssnitt) är en teknik för att definiera abstrakta typer som endast innehåller abstrakta metoder samt konstanter. Ett interface kan ej innehålla variabler. Interface definieras för att beskriva en uppsättning

metoder som en klass måste ha för att ha en viss egenskap. Skillnaden mellan ett interface och en klass är att klassen är en blandning av specifikation och implementation, medan ett interface endast är en specifikation. Ett interface kan alltså inte innehålla någon kod. För att applikationen skall kunna använda sig av ett interface så måste det deklareras tillsammans med klassen, ”***public class*** *Sms* ***implements*** *MapListener”*.

6.2.1 Inre klassen *AlarmListener*

En lyssnare skulle implementeras. Man kan i Java välja mellan att definiera en egen händelselyssnarklass som en separat klass i en egen fil eller att definiera den inuti den klass som skall användas. En inre klass är naturlig och praktisk eftersom händelselyssnaren endast skall användas inom denna klass. Applikationens inre klass, *AlarmListener*, är en händelselyssnare som ligger och lyssnar efter alarm som kommer från stacken.

6.2.2 Metoder

Metoderna i en klass beskriver de tjänster som klassen tillhandahåller. Metoderna innehåller den aktiva delen av klassen, den kod som verkligen utför något. Det finns i Java inget sätt att deklarera metoder eller funktioner utanför en klass, all aktiv kod som exekveras finns i metoder i klasser.

Följande metoder finns definierade i klassen *Sms.java*:

- *main()*

  I denna metod startas applikationen. Här anropas metoder för upp- och nedkoppling av stack och ROC. Här anropas även metoden som skall skicka SMS-meddelanden.

- *stackConnect()*

  Uppkoppling mot stacken sker i denna metod.

- *stackDisconnect()*

  Denna metod sköter nedkoppling av kommunikationen mellan applikation och stack.

- *processConfEvent()*

  Tar emot bekräftelse (*confirm*) på SMS-meddelandet.

- *processAlarmEvent()*

  Denna metod tar emot alarm från ROC och skickar dessa som SMS-meddelanden. För att kunna hålla reda på alla SMS-meddelanden (*request)* som skickats, sparas dessa i en länkad lista. Dessa meddelanden sparas tills man har fått en *confirm*.

7 Test

För att kontrollera om applikationen fungerar i miljön ”triggar” man felmeddelanden. Med ”triggar” menas att man tvingar fram specifika fel, man kan till exempel dra ur en sladd och se vad som händer. Vid uppkoppling av kommunikationen mellan applikationen och stacken används en manager. Denna manager arbetar i bakgrunden och i denna kan man ange vilka alarm som man vill ha. Varje felmeddelande har ett specifikt ID-nummer. Detta ID-nummer fås när felet inträffar. Man får sedan kontrollera vilket felmeddelande som motsvarar detta nummer. Testmiljön som beskrivs i detta kapitel utarbetades innan implementationen av applikationen var påbörjad. På grund av tidsbrist användes aldrig denna miljö.

8 Analys

Applikationen fungerar, dock i en begränsad omfattning. I nuläget fungerar applikationen för vissa felmeddelanden som skickas via SMS. Lösningen som har valts är på intet sätt den optimala lösningen. Det går alltid att göra en bättre lösning, men nivån på vår lösning har lagts på en nivå som ej överstiger vår kompetens.

8.1 Arbetets framskridande

Vi började med att utarbeta en tidsplan som vi skulle försöka följa. Första veckorna, när det var inläsning och viss design, gick detta mycket bra och vi låg bra till. Den miljö som behövdes för att vi skulle kunna bygga vår applikation drog ut på tiden. Detta gjorde att vi fick lägga oss på en sådan nivå att vi skulle hinna få klar en mycket enkel applikation. För att överhuvudtaget få en fungerande applikation fick vi bryta ner den i bitar och testa varje bit för sig. Detta gjordes på grund av att miljön byggdes på successivt.

8.2 Framtiden

Applikationen går att utveckla både grafiskt och kodmässigt. När ett felmeddelande uppstår skulle man till exempel kunna få fram meddelandet grafiskt på datorskärmen. I detta grafiska gränssnitt skulle man även kunna få fram en lista på inträffade fel, samt vilka som har åtgärdats. Felmeddelanden skulle även kunna skickas via email, vilket skulle ge en större valmöjlighet för support. På så vis skulle man kunna fördela belastningen, vissa mindre

viktiga felmeddelanden skulle kunna skickas via email. Felmeddelanden av typ *warning* och *notify* skickas dock via SMS.

9 Slutsats

Arbetet med detta projekt har varit väldigt intressant, roligt, utvecklande och lärorikt. De teoretiska och praktiska kunskaper som utbildningen har givit omsattes nu i praktiken. Detta ansåg vi var mycket lärorikt. Vi hoppas att det arbete som vi har påbörjat kommer att vidareutvecklas, så att Ericsson Infotech kommer att ha nytta av denna applikation.
Vi har även lärt oss att man i arbetslivet är beroende av andra faktorer än bara den egna arbetsinsatsen. Man är ett team av människor som arbetar tillsammans och alla har sin uppgift att fylla.

Referenser

[1] Jens Landgré. Unit Design Specification – SS7 ROC Management Interface R3A, SS7 OAM API. 3/102 62-CAA 201 669 Uen, 2000.

[2] Gunnar Heine. GSM Networks: Protocols, Terminology, and Implementation, 1999:Artech House Publishers, Boston London.

[3] Bruce Eckel. Thinking In Java, 1998: Prentice Hall PTR Prentice Hall Inc.

[4] Hans-Erik Eriksson. Programutveckling med Java, 1997: Studentlitteratur.

[5] Ericsson Infotech Portable SS7 Product Description.

[6] MAP ETSI R4 ETSI 08/96 Functional Specification API, 2000

[7] http://bitweb.bit.se/bitonline/1999/07/17/19990716BIT00280/07160028.htm

[8] http://www.bahnhof.se/~dc/sms.htm

[9] MAP ETSI R4 ETSI 08/96 – Functional Specification, Dialogues and Sequence Diagrams.

[10] http://justitie.regeringen.se/propositionermm/sou/pdf/sou97_49.pdf

A Ordlista

ANSI	American National Standards Institute
API	Application Programming Interface
GSM	Global System for Mobile Communication
GT	Global Title
INAP	Intelligent Network Application Part
ISDN	Integrated Services Digital Network
ISUP	ISDN User Part
JDK	Java Development Kit
MAP	Mobile Application Part
MTP	Message Transfer Part
OAM	Operation Administration and Maintenance
RAM	Random Access Memory
ROC	Reusable OAM Component
SCCP	Signaling Connection Control Part
SCP	Service Control Point
SDP	Service Data Point
SMS	Short Message Service
SS7	Signaling System No. 7
SSN	Subsystem Number
SSP	Service Switching Point
TCAP	Transaction Capabilities Application Part
TUP	Telephone User Part