Design of a Network Independent Emergency Service

Golara Khayltash

A thesis submitted to the Faculty of Engineering, University of the Witwatersrand, Johannesburg, in partial fulfilment of the requirements for the degree of Master of Science in Engineering.

Johannesburg, March 2006

Declaration

I declare that this thesis is my own, unaided work, except where otherwise acknowledged. It is being submitted for the degree of Master of Science in Engineering in the University of the Witwatersrand, Johannesburg. It has not been submitted before for any degree or examination in any other university.

Signed this ___ day of _____ 20___

_____

Golara Khayltash.

Abstract

Emergency services are vital for the minimization of damage, injury and loss of life. These services are, by definition, a combination of telecommunications and information services, and are by nature, distributed. However, most current emergency services do not take advantage of emerging technology, and hence, are restricted in the functionality they offer.

This project proposes the design a full information structure for an emergency call centre service, which can be offered as a service or application on any core network. As emergency services are distributed, and combine both telecommunications and information services, an appropriate design tool which caters for these issues, is the Reference Model for Open Distributed Processing (RM-ODP), which will be used in the design of the emergency service. In addition, OSA/Parlay Application Programming Interfaces (APIs) will be used for the application to access telecommunication network functionality.

The enterprise viewpoint examines the design requirements and considerations for an emergency system, which is the first step in designing a service based on the RM-ODP guidelines. Secondly, the information viewpoint is defined, which identifies the information flows between the objects and classes defined in the enterprise viewpoint with the aid of robustness diagrams and high level message sequence charts. Next, the computational viewpoint of the emergency service describes the components that the service consists of and the interfaces through which they communicate, enabling distribution of the system to be visualized. In addition, the engineering and technology viewpoints are briefly touched upon.

The RM-ODP proves to be a useful tool the design of this application. In addition, the use of OSA/Parlay APIs have also proved beneficial, enabling the application to run on any platform, irrespective of the level of functionality it already provides. The benefits that this design offers over conventional emergency services are allowing callers and emergency response personnel full access to the functionality of the

service, despite any limitations on their telecommunications network, finding the location of a caller from a fixed or mobile phone, ease and speed of obtaining relevant emergency information, and the ease and speed of sending relevant information to emergency response personnel.

Finally we recommend improvements in the reliability and accuracy of finding the location of mobile phones, as well as creating ways of identifying the location of VoIP users.

Acknowledgements

The following research was performed under the auspices of the Center for Telecommunications Access and Services (CeTAS) at the University of the Witwatersrand, Johannesburg, South Africa. This center is funded by Telkom SA Limited, Siemens Telecommunications and the Department of Trade and Industrys THRIP programme. This financial support was much appreciated.

I thank my supervisor Prof. Hu Hanrahan for his great support, guidance and encouragement, and Prof Rex van Olst for his assistance in the area of location based services, and his support and encouragement in general. I thank my colleagues at CeTAS for their valuable inputs and assistance during the duration of this research project, especially Rolan Christian and David Vannucci. Lastly, I thank my parents for all their support and encouragement, without which this project would not have taken place, and my sister, Shekufeh, brother-in-law Bayan, and niece, Arya, for their advice, assistance and encouragement.

Contents

List of Figures

List of Abbreviations

3GPP	Third Generation Partnership Project
ALI	Automatic Location Identification
ANI	Automatic Number Identification
API	Application Programming Interface
ASP	Application Service Provider
BTS	Base Transceiver Station
CLI	Caller Line ID
CORBA	Common Object Request Broker Architecture
CPE	Customer Premise Equipment
E112	Enhanced 112, used in Europe
E911	Enhanced 911, used in the United States
E-OTD	Enhanced Observed Time Difference
ECGI	Enhanced Cell Global ID
ETSI	European Telecommunications Standards Institute
FCC	Federal Communications Commission
GIS	Geographic Information Systems
GMLC	Gateway Mobile Location Centre
GPS	Global Positioning System
HLR	Home Location Register
IP	Internet Protocol
IN	Intelligent Network
ISO	International Standards Organization
LIF	Location Interoperation Forum
NGN	Next Generation Networks
ODP	Open Distributed Processing
OOP	Object Oriented Programming
OSA	Open Services Access
PSAP	Public Safety Answering Point
PSTN	Public Switched Telephone Network

QoS	Quality of Service
RM-ODP	Reference Model for Open Distributed Processing
SCF	Service Capability Feature
SCS	Service Capability Server
TDOA	Time Difference of Arrival
TOA	Time of Arrival
UML	Unified Modelling Language
VoIP	Voice over Internet Protocol

Chapter 1

Introduction

1.1 Importance of Emergency Services

Life in the 21st century, with the vast proliferation of technology, has brought us more efficient and faster ways of doing things, from manufacturing products in assembly lines, to leisure activities, to increased speed of travel, be it with car, high speed train, or airplane. Our lives have become much easier and more enjoyable because of these advances, but the risks we personally face on a daily basis have also increased drastically.

Thankfully, agencies responding to emergencies have also increased, providing us with access to the police, ambulance services, fire and rescue personnel, and the coast guard, just to mention a few. In addition, telecommunications has evolved over the years to provide the vast majority of people with access to a telephone, be it fixed line or mobile, facilitating access to these emergency services. However, the key to minimizing the amount of damage to person and property caused by emergencies remains in informing the relevant agencies of the details of the emergency as soon as possible, and in their timely response.

To address the issue of prompt response to emergencies, emergency service numbers, such as 112 for most of Europe, and 911 in Canada and the United States, have been devised, which can connect the user with a range of emergency response services, such as police, ambulance and fire response units. This is not a new development - Britain, for example, has had an emergency number since 1937 [4]. Additionally, legislation has been enacted in many countries, including South Africa, to establish a single number as a national emergency number, and to have it implemented throughout the country [5], [6].

Despite this emphasis on implementing emergency services nationally, Australia is one of the few countries, if not the only one, whose emergency number services cover the entire country, irrespective of the type of telecommunication used. (The United States, with its widely publicized 911 number, has about 75% coverage of its population [4].)

Generally speaking, there are two types of emergency services - one that deals with individual emergencies on a day-to-day basis, and another one, an emergency crisis response service, which acts to alleviate national or international emergencies, both natural and man-made, such as the events of Sept. 11, 2001, in the United States, or the October 2005 earthquake in Pakistan. Numerous organizations are involved in the organization of emergency crisis response services, both on a practical level, and on a more technical level, providing the necessary infrastructure to implement such widely deployed services. For instance, the Object Management Group has a task force devoted to software standards supporting emergency response and military operations [7]. This paper deals only with localized emergency services, and not the national response services.

1.2 Requirements for an Emergency Service

According to a survey made by Nextel [8], 75 countries throughout the world have single emergency numbers, with 112 being the most commonly used. Several other countries also have multiple emergency numbers, with different numbers being used depending on the nature of the emergency.

In most countries, there is specific legislation covering the use and implementation of these emergency numbers. For instance, the current model for 911 services in the United States specifies that:

- the numbers should be touch-tone generated
- incoming calls will be forwarded from the local or end switch to a PSAP (public safety answering point) responsible for the caller's location area
- the caller's location should be transmitted automatically to the PSAP using available resources, usually in a packet switched manner.
- calls need to be connected to the PSAP within 1.2 seconds of the last digit being dialed

One addition made for all European countries states that

- "calling 112 is free for the end user, be it a fixed or mobile call." [9]

Further legislation for wireless and mobile communication, as well as more sophisticated information transfer, caused by the advent of mobile communications, and, more recently, voice over IP, is in the process of being considered and implemented [10]. Enhanced 911 (E911) is an example of legislation that has been approved, and is in the process of being implemented. E911 refers to "an emergency telephone system which includes network switching, database and Customer Premise Equipment (CPE) elements capable of providing selective routing, selective transfer, fixed transfer, ANI (automatic number identification) and ALI (automatic location identification) [11]."

South Africa is at present working on the establishment of a national network of public emergency communications centres, known as 112 Emergency Centres. According to the Telecommunications Amendment Act of 2001;

> *The establishment of public emergency communications centres (112 Emergency Centres) to promote the health, safety and security of South Africans. 112 has been designated as our national public emergency number. The authority must implement this decision and develop and apply common technical standards and operating procedures for the relevant operators* [6].

These emergency centres are not envisioned to actually provide services during emergencies. Rather, their duty is to transmit any telecommunication request for an emergency service to the relevant emergency organization, as well as providing location information through voice, data and global positioning systems [12].

1.3 Traditional implementation of Emergency Services

Emergency services have, until now, been executed using the Public Switched Telephone Network (PSTN), with an Intelligent Network (IN)[1] overlay allowing related services to be implemented, such as call forwarding and reverse charge billing.

When a caller using a fixed line dials an emergency number, the closest switch or exchange recognizes the number as indicating that this call will be dealt with

[1]The PSTN and IN will be further explained in **Sect. 2.1**

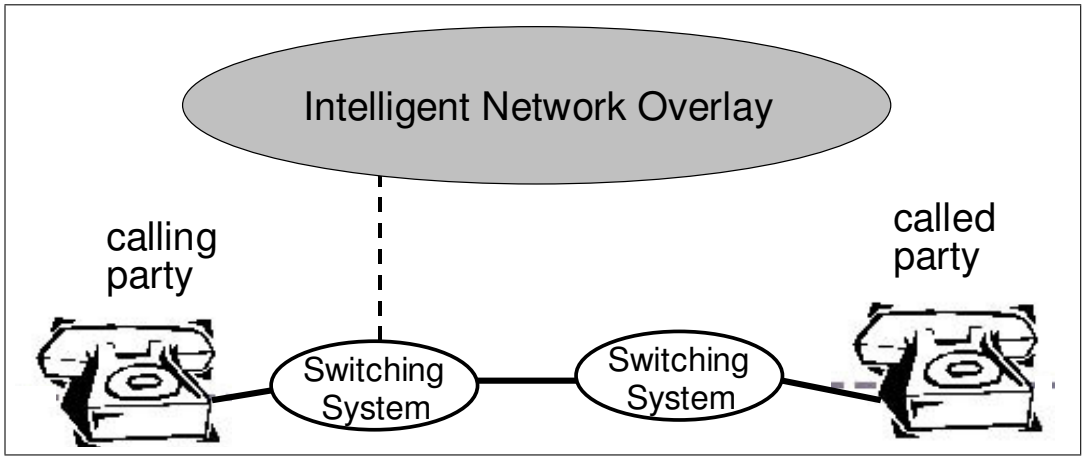

Figure 1.1: PSTN with IN overlay for services

differently. The routing tables present in the switch indicate which emergency centre the call is routed to. In the past, such calls were often routed over special dedicated circuits, to ensure the call went through, despite possible overloading on the network. With electronic control being introduced in the PSTN, these calls are sometimes transmitted on the same networks as other calls, but they still may be able to access circuits that other traffic cannot.

After the call is routed to an emergency centre, the call needs to be answered in a specified amount of time, depending on the legislation in that country. An emergency service dispatcher answers, and based on the nature of the emergency, either forwards the call to the appropriate emergency service, or dispatches the appropriate help to the caller.

The switch then forwards the relevant call information up to the IN layer, as shown in **Figure 1**, which then takes care of reverse billing and other services related to the call.

In many parts of the world, the emergency service can identify the telephone number of the caller, which is useful in cases where the caller can not give the location, or is unable to complete the call. This is usually done with the system the telephone company uses to bill calls, which makes numbers visible, even for people using unlisted numbers or who block the caller ID function. For a fixed landline phone, the caller's number is associated with their physical address, which gives the location the call is coming from.

1.4 Current Developments in Emergency Services

1.4.1 Locating mobile callers

The PSTN, which is described in **Section 2.1**, is the telephone network we all grew up with, and has, until very recently, been the main, and for the most part, the only telecommunications network in use. As an established telephone network, it remained unchanged for decades, and the way emergency services were offered on it also largely remained unchanged. However, in the '90s, with the introduction and proliferation of mobile phones, the implementation of emergency services and the nature of calls made to the emergency number started to change. The fact that mobile phones were not limited to a specific location made them far more useful than fixed lines in reporting emergencies, since the phones accompanied the user, and were readily accessible whenever an emergency occurred, either to the user or to someone else. The percentage of emergency calls being made from mobile phones increased, until well over half the total calls were being made from mobiles. In some European countries, close to 70% of emergency calls originate from mobiles [9].

However, this increase posed a problem. While the location of a caller from a land line was found relatively easily, that of a caller on a mobile was far more difficult to ascertain, because of the obvious mobility of the phone. Often times, even the caller could not be sure of his or her location, for instance, if the caller drove by an accident on a stretch of highway. According to statistics, in at least 6% of mobile emergency calls, the caller can not specify the location they are calling from [9].

As the percentage of emergency calls dialed from mobile phones increased, some countries, while trying to increase coverage of the service to all of their inhabitants, as well as increasing and upgrading the quality of the service, implemented methods to discover the location of mobile phones. Eventually, the government stepped in, in the United States and Europe, and passed legislation that required the mobile operators to first provide a call back number for the caller, and then to transmit the location of the caller to the PSAP.

In the United States, the Federal Communications Commission (FCC) first required all 911 calls to be routed and delivered to a PSAP by 1 October 1997, even if they were initiated from a mobile that was not currently subscribed to any network, or had blocked outgoing calls. Any phone that had the capacity to place an outgoing call would have a a call made to an emergency number routed to a PSAP. By 1 April 1998, the mobile operators were required to provide a call back number and location

data to the PSAP, and by 1 October 2001, the mobile operators were required to provide location data to within 125 meters, 67% of the time [13]. This is part of the United States' E-911 initiative. Similarly, the European Parliament, which also has an E-112 initiative, issued a directive making the forwarding of caller location by operators obligatory, in March 2002 [9].

The solutions which are available and most commonly used for identifying the location of a mobile handset are based on Cell ID [2], time difference of arrival (TDOA) [3] and enhanced cell global identity (E-CGI) [4]. Technologies that are available but not yet widely used, such as assisted global positioning system (assisted GPS) [5] and enhanced observed time difference (E-OTD) [6] are expected to give a more accurate position for the location, but will need a significant investment in infrastructure and handsets. As location based services for mobile communication networks are a rapidly growing industry, the scope for using these services to enhance the implementation of emergency calls increases greatly. However, in order to take advantage of this technology, PSAPs will have to upgrade their systems to be able to receive, process and use location information [9].

One obstacle encountered was the concern of mobile users that their location would be monitored, resulting in a lack of privacy. Legislation has been introduced in Europe, stating that the right to life and health take precedence over privacy, and that technical solutions for location identification must meet requirements for privacy protection.

1.4.2 Interfaces

There has also been some work done in standardizing interfaces, primarily the one between the Gateway Mobile Location Centre (GMLC) and the application, in this case, the emergency service. This interface, called "Le interface", governs the transfer of location information, adapting it to the specific application, and has been

[2] The mobile device is located using the location of the base transceiver station (BTS) with which it is communicating.

[3] The location of the handset can be determined from calculations using the difference between time of arrival (TOA) of the signal from a minimum of three BTS's

[4] E-CGI uses both Cell ID and the time of arrival to refine the location of the mobile

[5] Assisted GPS combines GPS, which uses a network of satellites to triangulate the receivers position and provide latitude and longitude coordinates, with information from the cellular network to obtain the location of the mobile

[6] E-OTD triangulates the position of the mobile from the coordinates of the BTSs the mobile communicates with, the arrival time of a burst of information from each BTS and the timing differences between the BTSs.

formally adopted by the Third Generation Partnership Project (3GPP)[9]. In addition, there is a strong drive within the industry in Europe to implement a common standard between a mobile operator and the PSAP, as well as between the PSAP and the emergency personnel. Standardizing these interfaces will ensure greater efficiency in the implementation of emergency services by minimizing cost, and embedding the functionality needed by the service in one interface.

1.4.3 Voice over IP

After the explosion of mobile phones onto the market, the next major change in the landscape of telecommunications has been Voice over IP, which uses computer lines to transfer voice over the internet, using the Internet Protocol (IP) signalling standard. There are two ways in which this is done, one in which the operator adopts VoIP as the bearer network, and the other, when the telephone user connects directly to the internet to transmit the call. In the first case, the operator will have full bearer and signalling compatibility with the PSTN, and location identification will not be an issue.

In the second case, however, telephones are seen as IP addresses over the internet, they appear as computers, and do not generally have locations assigned to them. This makes it difficult to geographically locate network users. In addition, since calls made over these phones are made on a different network than conventional fixed and mobile phones, emergency calls cannot be easily routed to the PSAP, and are actually impossible to route on some VoIP systems. Furthermore, in the event that a caller is unable to give a location, there is no other way emergency services can locate them.

Similar to initial efforts made by mobile phone carriers, several VoIP carriers are already implementing a technical way to work around this issue. The United States government has also set a deadline, requiring VoIP carriers to implement E-911, but due to the technical difficulties in doing so, the deadline is being appealed by several VoIP companies [14]. The Canadian government and European Union have also implemented legislation requiring VoIP service providers to offer the same level of 911 or 112 emergency service as is found on other telephone networks. How and when this will be implemented remains to be seen.

1.5 Critical Issues Facing Emergency Services

There are some serious issues facing the successful functioning of emergency services. Perhaps the most major technical problem centers on obtaining location information from mobile phones, with impending legislation deadlines compounding the issue by adding the need for obtaining the location of a VoIP caller. At the moment, especially in the United States, many PSAPs are unable to process location information from mobile networks, and as a result, will need to upgrade their facilities. Furthermore, mainly in some European countries, despite legislation, Emergency Authorities, as the PSAPs are called there, are unable to request the address details of a fixed call, despite decrees stating that network operators must facilitate access by the 112 emergency call service to the relevant databases, to obtain caller location information and other subscriber information [9].

Linked to this issue of call centers being unable to request location information from an external database is the lack of smooth interworking between various systems, caused mainly by non-compatible interfaces and formats, which ultimately result in critical information not being available when needed.

All these issues result in the delay of getting vital assistance to people in emergencies, putting their survival at even more risk. Time is of the essence in the successful handling of emergencies, time that should not be wasted in obtaining and confirming location and other information that can be accessed from databases and through other means.

1.6 Objectives of the Research

With the opening up of the telecommunications network, it is now possible for third party providers to offer services using telcos. In the past, the telco infrastructure was closed, with no provisions for any party other than the network operator to provide any outside services or applications. The same party was responsible for connectivity, transmission and service provisions. With the advent of open distributed services, networks have opened up in terms of topology, platforms and evolutions. Applications, network control and applications have been separated, making it possible for different parties to provide different services.

One such service that not only provides a useful service, but also is actually vital to the continued well-being of customers, is an emergency service, whose primary

benefit is saving lives and giving members of the population an increased sense of security, and whose implementation will also facilitate cooperation between existing emergency services. As technology has progressed further, especially in the area of telecommunications and information technology, we ask ourselves if these advances can improve the implementation and delivery of emergency services in any way.

This project proposes the design a full information structure for an emergency call centre service, which can be offered as a service or application on any core network, working towards the design of a national emergency call centre infrastructure, with the emphasis being mainly in the RM-ODP enterprise and information layer. This infrastructure will not be vendor specific, but can be implemented on any architecture or core network, and will assist towards the realization of a single emergency number accessible to all, that will be able to respond to any emergency. The Parlay information model, in the form of Parlay APIs, will be used in the information layer design to illustrate the various classes and the flows of information between them.

In the previous models of emergency services, since the functionality of the service resided in the network as an IN overlay, the features offered by the service were limited by the functionality of the network. However, in the new model of open service architecture, the different components of the service lie in the Service Capability Servers, which are separate from the network, and can be accessed by users of any network. This enables the service to offer far more functionality to the user, despite any technical limitations of the network being used, for instance, the inability to deal with location information, as mentioned in the previous section.

A service as crucial as an emergency service needs to be as up to date as possible, and use all available technology to improve its functionality. Creating an emergency service on open service architecture will streamline the service and enable it to rapidly access vital information, such as location and medical records, making it more efficient, faster, and most importantly, able to save more lives.

In light of the new ways services can be offered through open distributed systems, and the necessity of having efficient and reliable emergency services, this paper examines an emergency service of the future.

1.7 Outline of Report

The rest of this report is organized as follows:

Chapter Two examines the literature and describe the current state of telecommunications, as well as the new changes that are taking place, such as the move towards open distributed systems and convergence between engineering and software design. In addition, it reviews some tools used in the design of the emergency service.

Chapter Three considers the design requirements and considerations for an emergency system, reviews and describes the design methodology proposed by the RM-ODP, and examines the enterprise viewpoint of the service, which is the first step in designing a service based on the RM-ODP guidelines.

Chapter Four further examines the design of the emergency service by investigating the information viewpoint, which defines the information flows between the objects and classes defined in Chapter Two with the aid of robustness diagrams and high level message sequence charts.

Chapter Five looks at the computational viewpoint of the emergency service, which illustrates the components that the service consists of and the interfaces through which they communicate, enabling distribution of the system to be visualized, as well as briefly examining the engineering and technology viewpoints.

Chapter Six summarizes the work that has been done and draws conclusions from the research.

Chapter 2

Literature Review

2.1 Current State of Telecommunications

Until very recently, the PSTN has been the main telecommunications network in use. It has been only the introduction of mobile phones that has changed the telecommunications landscape. Based on circuit switching, the PSTN is composed of a series of switches that route calls locally, nationally, and internationally from one fixed land line to another. Value added services have been added using the IN, which adds functionality in the switches to provide primarily voice centric services. However, simple data services, like fax and email also operate over the PSTN.

Due to the nature of the PSTN, emergency and other services could only be developed and offered by the telco, keeping in mind the structure and architecture of the network. Since the intelligence for added services resides in the switches, only specialized telecommunications experts would be able to program the switches to implement a new service, resulting in very few people having the ability to create and offer services on the PSTN. Furthermore, every time a new service was to be implemented, the software running the service would have to be added to a great number of switches, making this process very long and tedious.

Therefore, traditionally, telecommunication services have been closed, with all services and applications being the sole domain of the network provider, since the telco's service logic was inaccessible from outside the network. Applications and value-added services were created and implemented using the Intelligent Network (IN), by a small number of skilled telecommunications experts, and were implemented entirely within the telco's domain. The IN enabled many services to be added on the PSTN and later, mobile networks, to a basic telephone call, ranging

from simple services such as call forwarding and reverse charging to more complex services such as conference calls and internetwork free phone.

However, since services offered over the IN need to be standardized before being implemented, the time-to-market of these services was quite long, and it was very difficult to devise and implement services over a short time. Because of the lengthy service creation process, only services that were guaranteed to succeed, and thought to be used by a large section of the population were developed, leaving no scope for exploring and experimenting with the viability of niche or short term services.

Furthermore, with the emergence of mobility and next generation networks (which is a multi-service bearer network capable of supporting multi-party, multimedia, real-time and information services, described further in **Section 2.3**) with telephony over IP, the communication network was no longer based on a rigid and highly formalized infrastructure. It became desirable for both third parties to be able to rapidly create many innovative applications, and to implement them without the long waiting processes for formal standardization.

2.2 Open Distributed Systems

Over the last several years, there has been a great deal of effort made towards "opening" up the network, especially for application development, which would entail third party providers offering value added services that could be implemented in a secure way over a telco's infrastructure. Networks are moving towards an open service architecture, which supports systems consisting of collections of independent computers that coordinate activities, share system resources, and appear to the users of the system as a single, integrated computing facility [15].

Open distributed systems are made up of components that may be obtained from a number of different sources, but together work as a single distributed system. They are basically "open" in terms of their topology, platform and evolution. They run on networks that are continuously changing and expanding, they are built on top of a heterogeneous platform of hardware and software pieces, and their requirements are continuously evolving [16].

This open architecture enables third party providers to provide services that will be able to access core network functionality through APIs (application programming interfaces), and will be applicable irrespective of the core technologies being implemented on a specific network. This means that applications can run on and migrate

to any core network implementation. Based on this, services and applications can be created and deployed rapidly, creating not only the freedom to experiment with the viability of various services, but also providing a means for rapidly creating necessary services, for example, an emergency service. No longer is service provision the sole domain of the network operator.

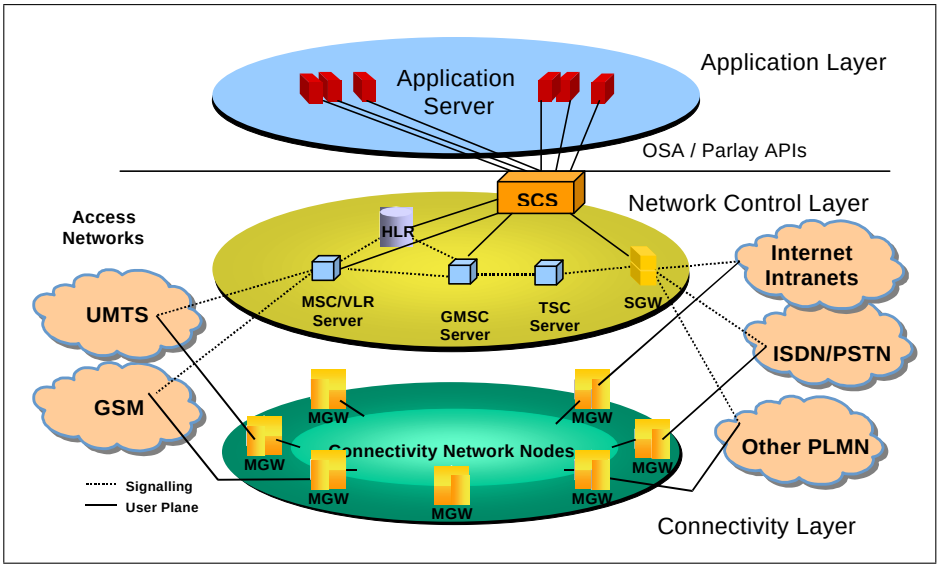

Figure 2.1: Open Service Architecture [1]

In an open distributed architecture, which is illustrated above in **Figure 2.1** the applications are logically found in the upper layer, which is called the Application Layer [1]. The layers below the Application Layer form the Service Control Layer, in which the Service Capability Servers reside. These Service Capability Servers each contain one aspect of the necessary functionality to handle calls, for example call control or charging. Below that is the Network Resource layer or the core layer, which controls the routing and distribution of calls. The application layer is based on open distributed technology, so applications can access core network functionality through open and standardized APIs.

Taking a closer look in **Figure 2.2** below, applications, - for instance, an emergency service - that reside on application servers are implemented by Service Capability Servers (SCS), which are nodes found on the border between the Application and the core network. These SCSs implement Service Capability Features (SCFs) which in turn contain the functionality for controlling calls, and potentially interact with the core network. The APIs, which form the service capability servers, should be network independent, so that applications do not have to rely on network details. It also enables easy migration and portability of applications to other networks, including legacy networks.

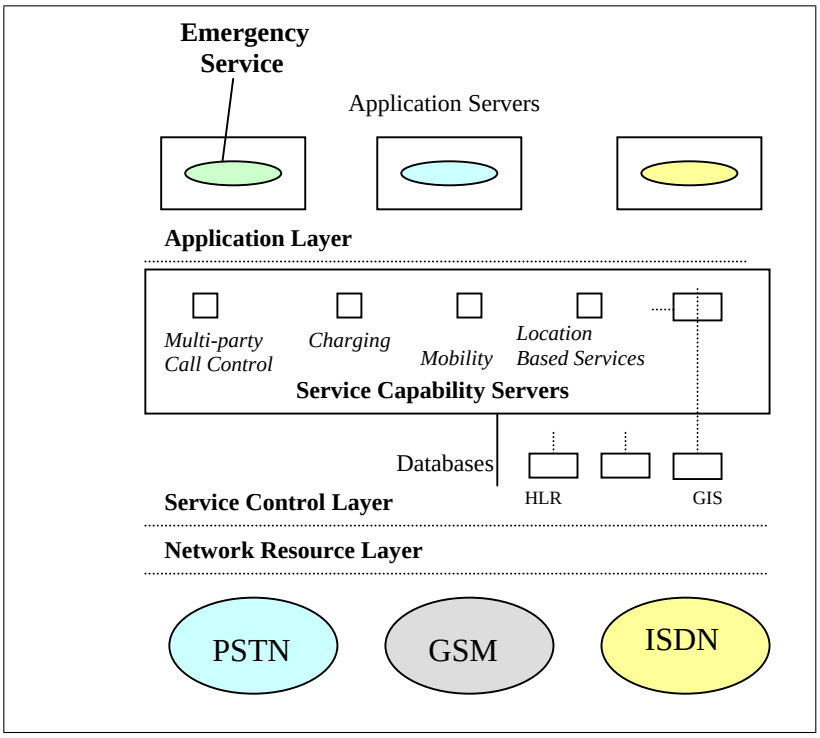

Figure 2.2: Detailed Overview of Open Service Architecture

In addition to implementing the APIs, the SCFs provide an interface to GIS information. Network related data, for instance, the location of telephones, or information stored in a mobile database - if the caller is calling from a mobile phone - will reside on databases. The SCF will provide a link between these databases and the service, by accessing the necessary information in a transparent manner.

In an emergency service, the associated APIs address user interaction - playing announcements to and obtaining information from the caller, user location - obtaining the location and caller identification of the caller, and charging the service instead of the caller, to mention a few.

In recent telecommunications standards, there is growing mention of emergency services, and how the standards must incorporate these services. Standards affect implementation, which can be seen in the OSA/Parlay Mobility API, for instance, where interaction with an emergency service is built into its functionality [17]. This clearly illustrates the acknowledged need among standard developers for emergency services built on emerging technology.

Another way of looking at open distributed systems is through the move from centralized intelligence to distributed intelligence in communications. Whereas before, in the era of the traditional telephone exchanges, intelligence sat on the SCF in the

IN or on the switch, now, with open distributed systems, intelligence has moved to the internet, where intelligence is completely distributed, and any person can mount an application server over the internet. Seeing how traditional telecoms and the internet is merging, it is far more beneficial to have intelligence distributed throughout the network.

With the advent of open distributed systems, opportunities are opening up for third party providers to provide telecommunications services. Along with this move towards opening up the telecommunication networks for application providers, another change has been taking place that greatly facilitates the provision of services for telecommunications - that of convergence between telecomms and information technology.

2.3 Convergence between Engineering and Software Engineering

With the rapid explosion of the Internet, many tools that were previously exclusively in the domain of telecommunication companies are now offered over the Internet. Indeed, as time goes by, there is more and more convergence between the world of computers and that of telecommunications, which has lead to next generation networks, the next step in telecommunications. The next generation vision of telecommunications incorporates aspects of both traditional telephony and the internet, and for this reason, needs a model that accommodates both centralized and distributed network intelligence.

The convergence of fixed, mobile and packet based networks has resulted in the Next Generation Network (NGN), which is a multi-service bearer network capable of supporting multi-party, multimedia, real-time and information services. One main objective of the NGN is the delivery of these services across various network infrastructures. The NGN is designed to facilitate creation of a wide range of advanced telecommunications and information services in a fast yet efficient way, taking the provision of services out of the hands of a select few telecommunications experts, and placing it in the hands of anyone familiar with computer programming.

Success of the NGN depends on the telcos utilizing multiple third party service providers outside the telco's domain, who will offer services that will use the telco's network. The services, of which an emergency service can be one, are provided by third party Application Service Providers (ASPs) above the telco's physical layer

(see **Figure 2.2**). Therefore, it is only natural that the tools used in present day telecommunications be a combination of both traditional tools and those used in software engineering.

2.4 Design Considerations

The nature of an emergency service is different from most other applications in several respects. Because of its urgent character, the service needs to be extremely robust, and cannot afford any down time. In addition, it needs to be absolutely secure, since it accesses, uses and produces very sensitive information, like medical records and personal information. Finally, the design methodology used in creating an emergency service needs to cope with both the telecommunication world as well as the software engineering world, since the functionality of the service is through telecommunications, but the software and databases that it needs to function are in the information technology world. Lastly, records of emergencies and related information need to be saved in a manner that can be accessed by the necessary organizations at a later time, for example, the police.

2.5 Aspects of Service Creation

Broadly speaking, emergency services lie in both the telecommunications and information systems worlds. They not only include communications, as well as having to meet statutory and functional requirements in the telecommunications industry, but are also distributed by nature, and rely heavily on the collection, storage and dissemination of information. In an emergency service, the information services aspect relies on the telecommunications part for functionality.

An emergency service is potentially complex and needs to be specified from a top-down perspective. An object-oriented approach is well suited to this approach, in which the most fundamental objects are first defined, and other objects inherit from them to add more detail. Object orientation (OO) also enables boundaries between systems to be clearly identified. Interworking can then be encouraged, monitored or prevented at system boundaries [18]. This is very much an RM-ODP approach, which adds to the usefulness of using it in modelling this service.

In OOP, software objects are used to model the real-world objects we find around us which are involved in a service or computer program. The computer program

will then be seen as a collection of units or objects that act on each other, and are capable of receiving messages, processing data and sending messages to other objects, as opposed to the traditional view of a program as a list of functions or procedures. Modelling a program after reality gives it simplicity, more flexibility, makes changes easier to implement, and in general, makes the entire program easier to develop and maintain. If, for instance, you have a program to record sales at 3 chain stores, the addition of a new store will entail the addition of a single object representing that store, instead of lengthy changes made to the commands and procedures [19].

Modelling using UML (unified modelling language) [20] finds its place in the RM-ODP and continues the OO paradigm in helping to identify the various players and domains involved in an emergency service, how they interact with each other, and how information will be handled by the system. These will all be illustrated in **Chapter 3** as part of the enterprise design of the emergency service.

Because of these reasons, the RM-ODP has been chosen as an approach to designing an emergency service, since it integrates the two aspects of telecommunications and information services into one distributed framework, and through its object oriented approach, decreases the overall complexity of creating the service. The next section will describe in more detail the viewpoints of the RM-ODP that facilitate service design.

2.5.1 Parallels between the Reference Model for Open Distributed Processing and Software Engineering

The RM-ODP uses five viewpoints to describe ODP systems. To simplify the vast amount of information associated with each system, these viewpoints each look at a certain aspect of a system, at a different level of complexity. The enterprise viewpoint defines the purpose, scope and policies of a system at a very high and technology independent level. The information viewpoint examines the semantics of information and information processing between the objects; the computational viewpoint defines the execution of the process as a model of distributed processing, and describes the functional decomposition of objects, attributes and interaction of objects. The engineering viewpoint describes the infrastructure required to support distribution, and the technology viewpoint examines choices of technology for the implementation of the system.

Figure 2.3 shows how the RM-ODP viewpoints can be related to the software engineering process [21].

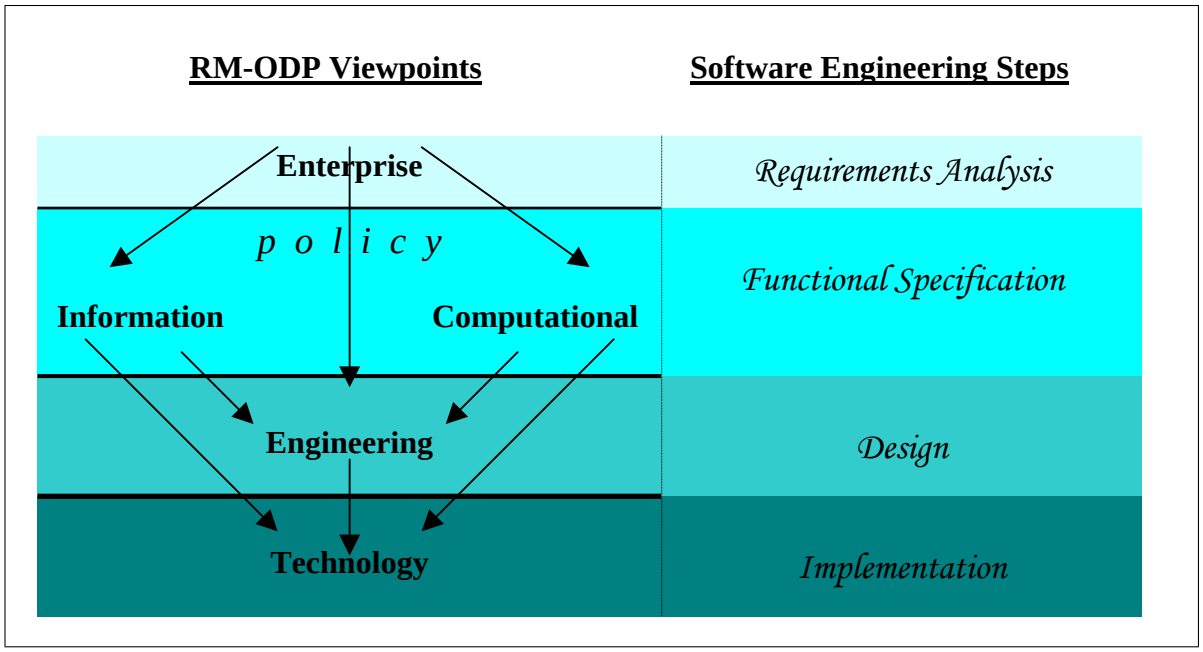

Figure 2.3: Correspondence between RM-ODP Viewpoints and Software Engineering

2.6 The Reference Model for Open Distributed Processing

In 1988 the International Standards Organization (ISO) began work on preparing standards for Open Distributed Processing (ODP). These standards define the interfaces and protocols to be used in the various components in an open distributed system. Prior to that, a reference model, the Reference Model for Open Distributed Processes (RM-ODP), which is a joint effort by the international standards bodies ISO and ITU-T, was developed to provide a coordinating framework for the standardization of open distributed processing. It creates an architecture that supports distribution, interworking, interoperability and portability in open distributed processes, and enables the construction of distributed system solutions that can cope with heterogeneity between systems [21],[22],[23].

The main objectives or aims of the ODP reference model are providing portability of applications across heterogeneous platforms, interworking between ODP platforms through meaningful exchange of information and functionality throughout the distributed system, and distribution transparency, or hiding of the consequences of distribution from both the applications programmer and the user [21].

The RM-ODP framework defines open distributed systems using five viewpoints or

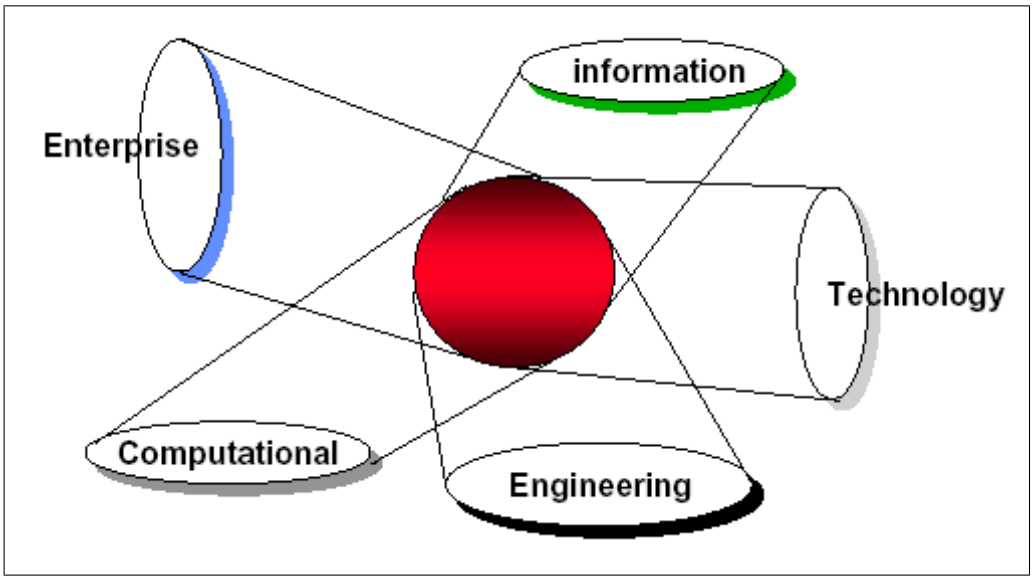

Figure 2.4: RM-ODP Viewpoints [2]

abstractions, the enterprise, information, computational, engineering and technology viewpoints, which are illustrated in **Figure 2.4** and briefly elaborated on.

- **Enterprise viewpoint -** describes the purpose of the enterprise, the scope and policy of the system, and defines rights and responsibilities, and their transfer. It creates a model of the system and its environment.
- **Information viewpoint -** provides a model for information analysis. It describes information required by the system and its meaning, as well as objects and their attributes.
- **Computational viewpoint -** defines the execution of the process as a model of distributed processing, and creates a model for distributed programming languages. It describes the functional decomposition of objects, attributes and how objects interact.
- **Engineering viewpoint -** describes the infrastructure required to support distribution, and the mechanisms which make the technology accessible.
- **Technology viewpoint -** describes the implemented system. It defines how a system is structured in terms of hardware and software components, and underlying supporting infrastructure.

The RM-ODP can be used to create an emergency service that can be offered to and implemented on various networks, irrespective of the type of technology used on

them. In addition, the open standards approach can be especially beneficial in a case such as emergency services, where different companies may have been contracted to provide emergency service centres in different states or provinces. Although the type of structure and technology used in the emergency centres may differ, using an open standards approach will enable interoperability between the different systems, and create a national system that will function and appear unified. The next section illustrates how OSA/Parlay assists in the creation of an application by third party providers.

2.7 Parlay and Service Provision

As mentioned in **Section 2.2**, telecommunications networks are opening up and allowing third party service providers to implement value added services on their networks. Of course, any method that would allow access to a telco's network would have to be extremely secure, and have provisions in place for authentication and authorization.

There is a range of standards in place to allow applications to access network functionality, the best known and most currently used of which are the OSA/Parlay standards. Instead of devising a new set of standards, we propose to use the OSA/Parlay standards in this project to enable the emergency service to be able to use the core network of whichever telecommunication network it will be implemented on.

Parlay is based on the open service access (OSA) architecture, which was explained in **Section 2.2**, and is illustrated in **Figure 2.5**. Parlay gateways are found between third party service applications and the telephone network infrastructure. Through the Parlay APIs, which are the units that allow access to the functionality of the network, these third party services can be run securely on any telecom operator's network, with full access to the network's functionality, independent of the implementation and protocols used in the underlying network. This eliminates the need to tailor make services and applications for each specific network and environment that it might be run on, and creates a single generic application that is able to run on any underlying technology, greatly simplifying the service creation process, and opening up applications to a much wider user base.

Parlay and its architecture is based on the same goals of combining the worlds of software engineering and telecommunications to rapidly create innovative applications that combine network features with Internet services and data as the RM-ODP,

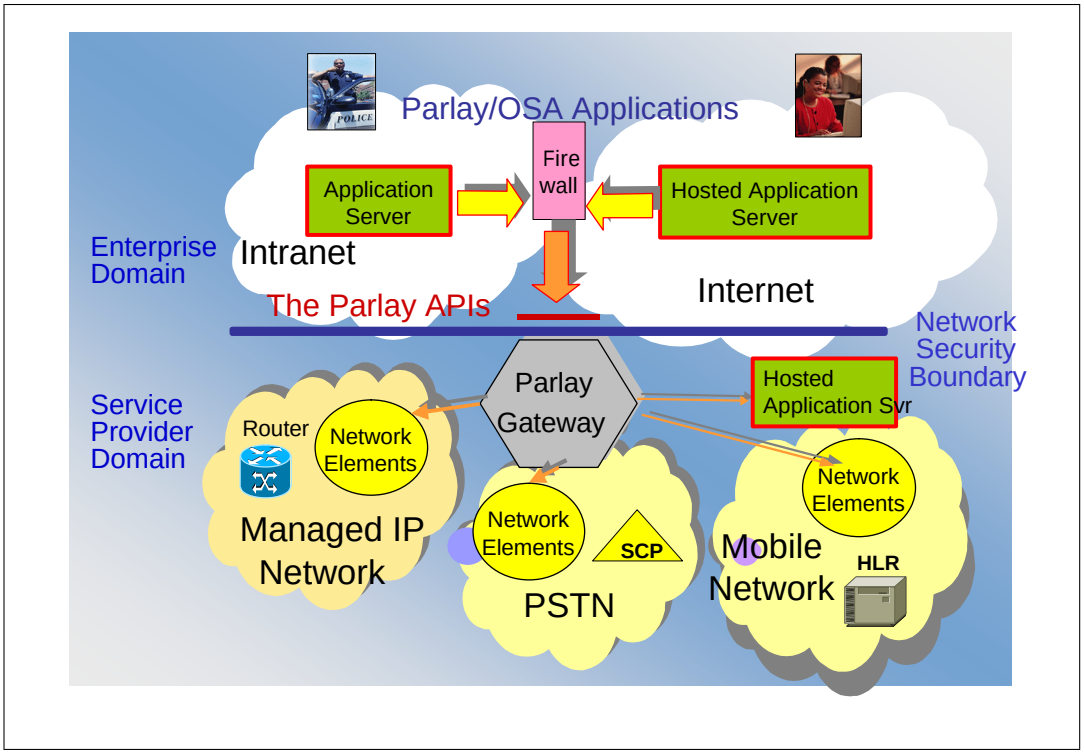

Figure 2.5: Parlay in the network [3]

which makes it a natural progression for implementation following the RM-ODP design process.

2.8 OSA/Parlay

OSA/Parlay defines "an architecture that enables service application developers to make use of network functionality through open standardized interfaces" [24]. "These standardized interfaces are defined using different APIs. Each Parlay API is logically contained within a Service Capability Feature (SCF) and each feature is housed within a Service Capability Server. Various SCFs are contained within a Parlay Gateway. [25]".

Each SCF allows operations to be carried out on specific network resources in a way that hides the underlying intricacy of the network. In addition, the SCFs act as an intermediary, delivering network events and responses to the applications. Grouping functionality into different SCFs allows developers to reuse the SCFs in creating applications that provide specific services to users.

2.8.1 Parlay APIs

> *The Parlay Group aims to create an explosion in the number of communication applications by specifying and promoting open Application Programming Interfaces (APIs) that intimately link IT applications with the capabilities of the communications world* [26].

As mentioned in the section above, Parlay APIs allow applications to access the functionality of telephone networks and generic support functions in a secure manner. The Parlay API specifications, which are open and technology independent, enable advanced telecommunications services to be provided by third party service providers. The APIs fall into two categories, framework interfaces and service interfaces.

The framework interfaces, through pre-defined classes, objects and methods, take care of authentication of the application, authentication for the application to use certain service capability features, discovery of network service capability features, establishment of a service agreement between the application and the network, and finally provide access to the network service capability features [27].

After the framework interfaces have taken care of the authorization and access, the service interfaces provide the applications with access to the network capabilities and information they require, ranging from call control and mobility to account management and multi-media messaging.

Apart from the **Framework** SCF, which is used for authentication of the service, service discovery and service selection, the other SCFs used by the emergency service are **Multi-Party Call Control**, **Mobility**, **Data Session Control** and **Charging**.

Chapter 3

Enterprise Viewpoint

3.1 Introduction

As described in **Chapter 2**, the enterprise viewpoint of the RM-ODP consists of a high-level, technology independent description of what the service does. All the various participants, or role players, in the service are listed, as well as what each of them can and can not do. In addition, it puts role players into natural groupings, called communities and federations, based on common purposes between the role players [28].

3.2 Description of Emergency Service

The infrastructure of the emergency service consists of local emergency centres located in every municipality. The staff at these centres are separate from emergency response personnel, and are responsible for call taking and dispatch of the appropriate emergency staff to the scene of the emergency. These emergency services are obligated to provide continuous and reliable service. Incoming calls will be routed to the emergency centre closest to the caller, and must be answered by an dispatcher within a specified number of rings, without busy tones or voice messages being played. After the location and information about the emergency is obtained, both from the caller and relevant databases, the dispatchers will dispatch relevant emergency personnel to the site, and send related information, such as the location and perhaps the medical records of people involved, through a data connection to the emergency personnel who are attending the call. When the emergency has been dealt with, a record of the call and emergency will be saved. These emergency calls

are free for the caller, and will be billed to the government agent.

In the case of people calling about routine or maintenance issues, like an electricity blackout or burst pipe, as soon as the operator has ascertained that the situation is not an emergency, the caller will be routed to an automated voice response system, which will give the numbers for the relevant municipality offices, and will free the lines for real emergency calls.

A trader function or broker is not included in this service, since the operator fulfils those functions.

3.3 Role Players in the Emergency Service

In the enterprise viewpoint of the emergency system, the role players are the:

1. **Caller** - any person dialling the emergency service number from any telephone or network to report an emergency.
2. **Telco** - telecommunication agent responsible for providing authorization and connectivity from caller to emergency service.
3. **Government agent** - organization responsible for commercially managing the services offered to the caller according to a contractual agreement.
4. **Service provider** - organization that offers the emergency service to the government agent.
5. **Emergency service** - technical infrastructure that enables callers to be answered by the operator, information to be gathered, verified, stored, and dispatched to relevant emergency response staff.
6. **Emergency service dispatcher** - emergency service staff, who receives emergency calls, collects information, and dispatches it to appropriate emergency response staff.
7. **Emergency response staff** - any person who receives emergency information from the emergency service dispatcher, and is tasked with responding to the emergency, for example, police, paramedic, fire and rescue services.

There can be many different business models for an emergency service; for instance, the telco could act as a service provider and government agent. This paper examines

the design of a possible emergency service without imposing any of the various models on it.

The policies that define the service are grouped into three areas by the RM-ODP: obligations, which are actions which the role players must fulful, permissions, which are the different options available to the role players, and prohibitions between the role players, the actions which they are not allowed to perform.

3.3.1 The Caller

The *obligations* of the caller consist of:

1. Providing the emergency service with accurate information relevant to an emergency.

The *permissions* of the caller consist of:

1. Unrestricted access to emergency service at all times.

The *prohibitions* of the caller consist of:

1. Contacting emergency response staff directly.

3.3.2 The Telco

The *obligations* of the telco consist of:

1. Authorization and connectivity provided for all users at any time from any telephone on any network, despite status of telephone, i.e. outgoing calls barred.
2. Routing provided for caller to the nearest available emergency service. In the case that the nearest emergency service lines are all busy, the call will be routed to the next closest emergency centre.
3. Reverse billing of all calls made to emergency service to the government agent.
4. Reserving lines especially for use by callers to emergency service, to ensure constant availability of service despite congestion on networks.

5. Providing pre-defined quality of service to all users of the service at all times.

3.3.3 The Emergency Service

The *obligations* of the emergency service consist of:

1. Providing the functionality to conduct multi-party calls between the caller, the emergency service operator and other parties.
2. Providing the functionality for efficient and reliable working of the emergency service by the emergency service operator.

The *prohibitions* of the emergency service consist of:

1. Discontinuation of the emergency service for any amount of time without providing automatic routing to the next nearest emergency service centre.

3.3.4 The Emergency Service Operator

The *obligations* of the emergency service operator consist of:

1. Response provided by emergency service operator within specified time period, i.e., five rings.
2. Prompt gathering and verification of caller information, location information and relevant history by emergency service operator through caller, and assisted by caller line identification (CLI), caller location identification, and associated databases.
3. Prompt dispatch of emergency information and location to closest appropriate emergency personnel by emergency service operator, i.e.; through transmitting a map of emergency location and relevant support information, like fire hydrant locations in the case of a fire.
4. Accurate logging of all relevant information and details by emergency service operator.

The *permissions* of the emergency service operator consist of:

1. Rerouting a caller to an automated voice response system after ascertaining the caller does not have a valid emergency.
2. Access to relevant databases not in the domain of the emergency service, like medical records and the Home Location Register (HLR) for mobile callers, by the emergency service.
3. Initiating multi-party calls between the caller, emergency service operator and emergency response staff.
4. Reporting a caller to the police if the information received from them has been deliberately false or misleading.

The *prohibitions* of the emergency service operator consist of:

1. Dissemination of gathered information to any third parties apart from emergency response staff and legal services.

3.3.5 The Emergency Response Staff

The *obligations* of the emergency response staff consist of:

1. Immediate verification of receipt of information from emergency service dispatcher.
2. Immediate dispatch of emergency response staff to the emergency location.
3. Providing assistance over the telephone through a multi-party call to the caller, if necessary, until emergency response staff has reached the scene.

The *prohibitions* of the emergency response staff consist of:

1. Emergency response staff contacting the caller directly, and not through the emergency service operator (to ensure gathering of all relevant information).

An example of an emergency service, and the interactions between the different role players, is illustrated in **Figure 3.1.**

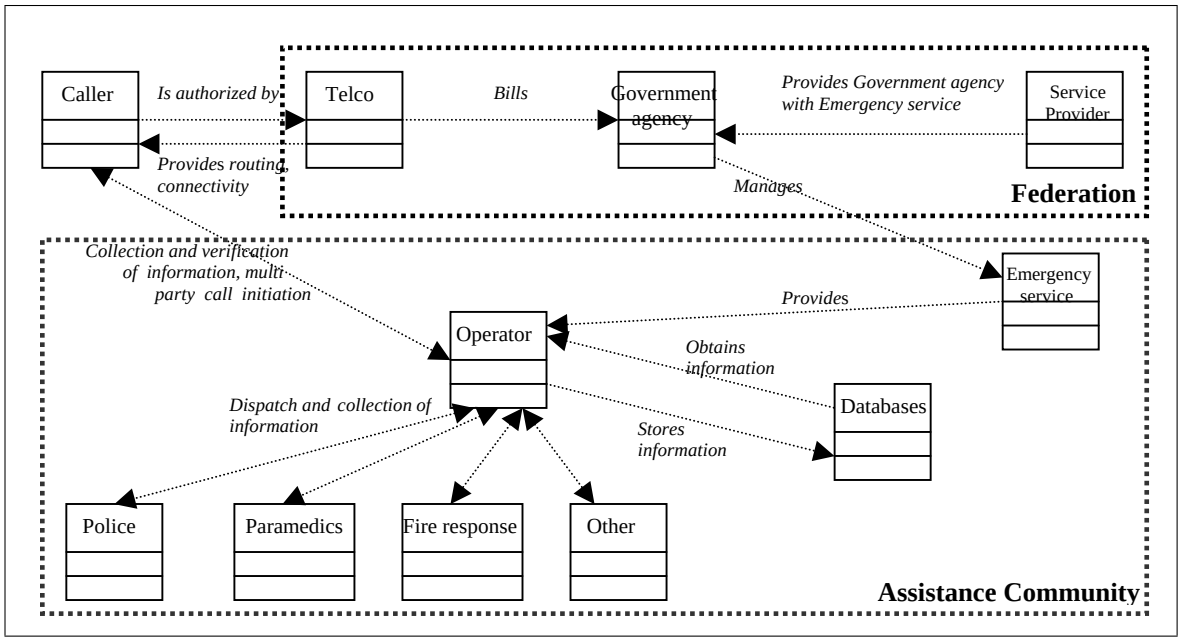

Figure 3.1: Overview of Role Players and Obligations in an Emergency Service

3.4 An Assistance Community

A part of the emergency system can be considered as a community, which is a grouping of objects or role players that achieve a common purpose. The emergency service, emergency service operator, emergency response staff and databases work as a unit to offer assistance to callers experiencing emergencies.

The resources within this community consist of information provided by the user, the databases, and the emergency response staff. The emergency service operator is responsible for the gathering of all the information, and its final storage in the relevant databases. The operator is responsible for providing information to the other parties within this community. In addition, the expertise provided by the emergency response staff forms another resource of the community. The operator is responsible for bringing the expertise of the emergency response staff to the relevant emergency, through dispatch of the staff to the appropriate location.

The members of this community are structured in a hierarchy, with the operator on the top level, and all the other members on equal ground, and answerable to the operator.

3.5 A Support Federation

A federation, which is a particular type of community in which a number of role players who answer to different authorities come together to achieve an objective, can also be formed among some role players. The telco, government agency and service provider form a federation which provides support and infrastructure for the emergency service to operate. Without the necessary telecommunications infrastructures and availability requirements, support, legislation and financial assistance from the government agency, and of course, the service provider, an emergency service would not be able to function. The relationships between the role players in this federation are covered in the obligations, permissions and prohibitions stated above.

The assistance community and support federation are both illustrated in **Figure 3.1**.

3.6 Security Issues

Security is a very important issue to manage, especially since much of the information involved in an emergency service is highly sensitive and personal by nature, and any errors introduced into the stored information, either accidentally or purposefully, can not only cut down on efficiency, but actually pose a life-threatening risk to the caller. As mentioned in the prohibitions of the emergency service operator, dissemination of any information gathered through the emergency service to an unauthorized third party is prohibited, and can be punished by law. Within the emergency service setup, the emergency service operator is the only one who is permitted to access and save information to the databases. Provisions do not exist for any of the other role players to access the databases.

Security breaches from outside the emergency service setup are also a possibility, especially since some of the databases used do not reside in the emergency services domain. Where possible, access to the databases shall be restricted to authorized staff upon entering the necessary passwords. Encryption will also be used where possible when retrieving information over distances. The telco and the owners of the databases are responsible for the security of the lines, and adequate protection of the databases.

3.7 Quality of Service

Yet another vital issue in an emergency service is the assurance of a pre-defined quality of service, to ensure legible speech and rapid transmission of information at all times. The government agent and telco will agree on the levels of Quality of Service (QoS) required, and the telco is responsible for its maintenance. In an emergency service, information saves lives, resources, and property, and any factors that impede the receipt and transmission of information can potentially cause harm. Both accuracy and speed of transmission are essential. Because of this, it is necessary that an excellent level of QoS is maintained, and measured to ensure compliance.

3.8 Contract

A key part of the enterprise level is a contract, which specifies the obligations of the various role players. In this model of an emergency system, one contract is made between the telco and government agency, which covers:

1. Required Quality of Service
2. Authorization, connectivity and routing to emergency service centers
3. Billing
4. Appropriate dimensioning of network to allow capacity for emergency calls

A contract between the service provider and the government agency would specify:

1. Reliability and security of emergency service infrastructure
2. The emergency service meeting legislative requirements
3. The emergency service meeting statutory and functional telecommunication requirements.

The details of the contract specifications are elaborated upon above under the policies of the role models.

3.9 Domain Model

As part of the enterprise view, we use OO tools to further refine the overall design, as mentioned in **Section 2.5**. Domain models are similar to the enterprise diagram in that they identify both the physical and abstract objects in the system, and show how they relate to each other. The objects in the model help to refine the classes that will be used in the actual programming of the service.

The following chapters will focus mainly on refining the design of the emergency service.

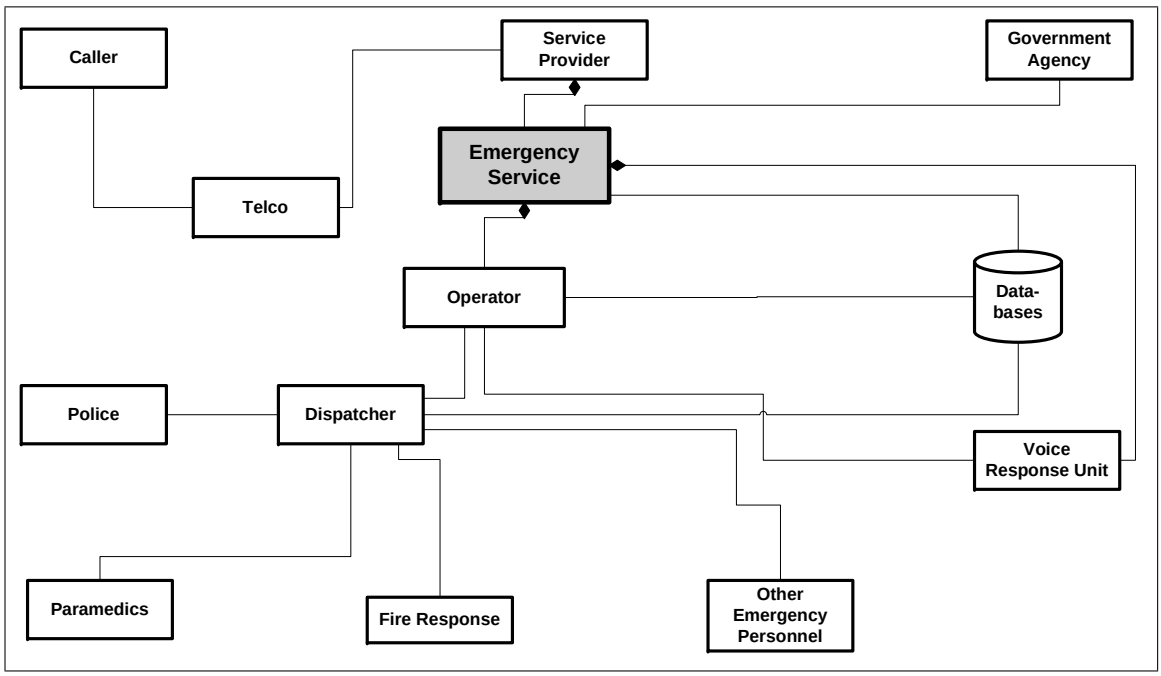

Figure 3.2: Domain Model of Emergency Service

Figure 3.2 shows the various objects identified at this early stage of the design process, and how they relate to each other. While the enterprise model shows the information flows between the classes, the domain model shows the relationship between them. A line with no arrows at either end between two objects means that one class *uses* the functionality in the other class, whereas a line with a diamond shaped end means that the one class *has* an instance of the second class. This can be illustrated in **Figure 3.2** by the line between the service provider and the emergency service, which shows that a service provider *has* an emergency service.

3.10 Use Cases

Following the domain model is the use case diagram, which analyzes the system and breaks it down into all the requirements and tasks that need to be performed by it. These actions are grouped into use cases that identify the interactions between the end user and the system. For instance, keying in a password would be too small a task for a use case. Rather, that would be included, with logging on, choosing an option from a menu, and the computer responding, in a "Select Service" use case.

The use case diagram then presents all the individual use cases that illustrate the requirements and actions of the service into a single diagram, and shows the associations between them. **Figure 3.3** illustrates the use case diagram for the emergency service.

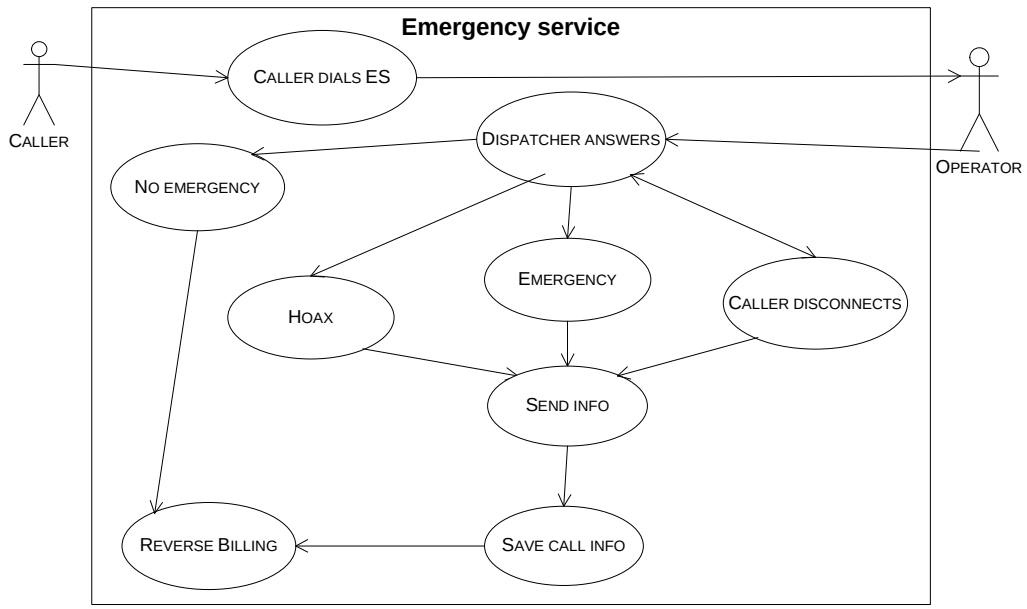

Figure 3.3: Use Case Model of Emergency Service

Many emergency services are implemented with a call taker, whose job it is to answer incoming calls, filter out the hoaxes and deal with the callers who don't have proper emergencies. The real emergencies are then forwarded to the dispatcher, a person with some degree of medical and first aid knowledge, who will then deal with the situation, or will forward the call again on to the proper emergency response personnel. This method of having both a call taker and a dispatcher reduces the number of skilled personnel needed in a call centre, since the more qualified dispatcher only deals with actual emergencies, which are usually only a fraction of the

total incoming calls. The disadvantage of this system, however, is that valuable time is spent in answering the call, ascertaining that it is an emergency, and passing the call to the dispatcher, valuable time that might make the difference in an emergency.

Initially, this emergency service was designed with both a call taker and a dispatcher. After going through the design process a few times, it seemed that the functionality of the service would not differ much if there was both a call taker and a dispatcher, or just a dispatcher, who would also answer the incoming calls. However, eliminating the call taker would make the design more streamlined, with only the essential players shown and illustrated. For this reason, in subsequent versions of the use case diagram, the call taker has been taken out, as illustrated in **Figure 3.3**.

Following the use case diagram, each use case is decomposed into the simple actions that it is composed of, leaving out responses from the system. The use cases divide the system into parts according to the ways that the users of the system interact with it, helping the OO approach by using this breakdown to organize modelling the service. Details about the objects in the system are discovered by using the use cases, first in writing them to define what the objects do, then by creating robustness diagrams from the use cases.

Experience has shown that the best way of writing use cases is in an active voice, driven from the point of view of the system user, and working inwards from the user interface to uncover the functionality required for a given usage scenario, and ultimately the objects that are needed to provide that functionality.

The use cases are listed below:

Use Case: Caller dials emergency service

- Caller dials emergency number from any telephone capable of making outgoing calls
- Call is routed to closest emergency centre

Use Case: Dispatcher answers

- Dispatcher answers within given number of rings and asks nature of emergency
- System automatically obtains the telephone number, location of caller, and name, where possible
- Caller describes situation to dispatcher

Use Case: Caller does not have an emergency

- Dispatcher provides information or redirects call to automated voice response unit or another number
- Call is disconnected

Use Case: Caller disconnects

- Caller hangs up or call gets disconnected before all details of emergency are reported
- Dispatcher calls the caller
- Caller answers
- Dispatcher asks for a detailed account of accident

  Alternate Flow: Caller does not answer

  - Caller does not answer the phone

Use Case: Hoax

- Dispatcher retrieves phone number from system
- If repeat offender, dispatcher sends information to police
- Dispatcher disconnects line
- Call Take is free

Use Case: Emergency

- Dispatcher asks for a detailed account of incident
- Dispatcher logs name and location of caller on screen, as well as any other info
- Dispatacher calls up medical records of people involved in emergency from database
- Dispatcher confirms location of caller on map

- Dispatcher sets up conference call between caller and ambulance, if necessary
- Dispatcher keeps caller on the line
- All necessary emergency response personnel confirm arrival at scene
- Call ends, dispatcher is free

Use Case: Dispatcher sends info to ES personnel

- Dispatcher sends accident information and map, if relevant, to ambulance, police and rescue personnel

Use Case: Dispatcher saves call info

- Dispatcher logs the name of the caller, phone number, and the incident, as well as background information.

Use Case: Billing

- Caller hangs up phone
- Billing information is debited from emergency service or govt. agency

3.11 Design Revisions

Design of anything is a fluid process, with changes and revisions presenting themselves throughout the process. This is the main reason there are so many tools for design, software engineering design being no exception. This is also the reason so much of this thesis is devoted to the design of the emergency service, instead of simply diving into the writing of the code - so that by the time the several stages of the design process are completed, the design will be finalized, and that the most time consuming part of all - the writing of the code - will be done in the most efficient manner, instead of having to be redone several times, due to unforseen design criteria changes. This process has been no exception, and some changes have been made from the original use cases presented above.

One final note on the order of the design steps: Depending on what literature one refers to, sometimes the domain model is listed as the first step, and other times

it is inserted after the use case diagrams. It seemed more intuitive to first identify the various classes, and then describe the actions between them, which is why this particular order has been chosen.

Chapter 4

Information Viewpoint

Now that we know the role players in the service, and what they can and can't do, the next step is to design the information viewpoint, in which the information that is passed between the role players is identified.

4.1 Robustness diagrams

After all the use cases are described, and one is sure that all the functionality of the system is captured by them, the next step in refining the design is in the form of a robustness diagram.

To create a robustness diagram, one works through the text of a use case, using the objects that have been discovered up to this point. This exercise helps to identify what necessary classes have been left out in the use cases, ensures a consistency of terminology with the other use cases that have previously been analyzed, and makes sure that the use cases are robust enough to represent the requirements of the system that is being built. They also identify possible objects to support the logic described in the use case, effectively acting as a bridge from the high level descriptions of the use cases to detailed design shown in the next steps of the design process, such as sequence diagrams. It is not easy to go from a high level analysis to a detailed design, which is why robustness diagrams are so important in the design process.

A robustness diagram provides a graphical view that illustrates the description found in a use case. Drawing the robustness diagram for a use case gives a visual completeness check that shows the entire use case has been accounted for. Each use case, as illustrated below, is illustrated by a robustness diagram, in which the objects, which

are run-time entities, are classified into boundary objects, control objects or entity objects. The actors are shown, similar to the use case diagrams. Next, boundary elements are identified, which form the boundaries of the system, and represent any sort of interface that actors interact with, such as screens, HTML pages or system interfaces. Then, the control elements are specified. As the name suggests, control elements represent objects that implement the logic required to manage the various elements and their interactions, such as a routing or alerting control element, shown in the first robustness diagram below. And finally come the entity elements, which represent the classes in the system, such as the emergency centre or database [29].

The robustness diagrams illustrated below deal with the information passing and processing in the emergency service, and this, help to refine the information viewpoint.

Use Case 1: Caller dials emergency service
Caller dials emergency number from any telephone capable of making outgoing calls, call is routed to closest emergency centre, dispatcher answers within given number of rings

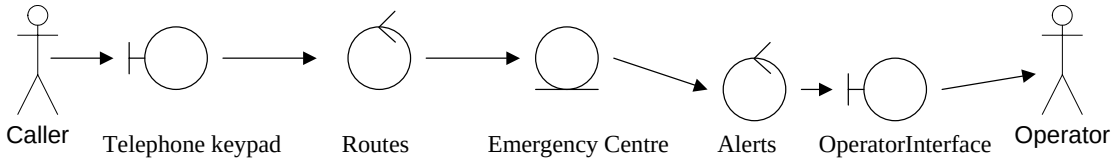

Figure 4.1: Robustness Diagram: Caller dials emergency service

Use Case 2: Dispatcher answers
Dispatcher answers within given number of rings and asks nature of emergency, System automatically obtains the telephone number, location of caller, and name, where possible, caller describes situation to dispatcher.

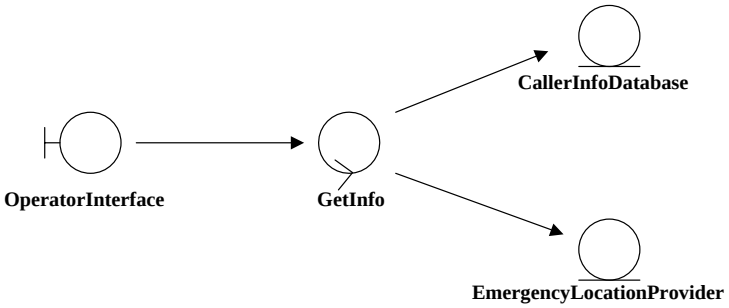

Figure 4.2: Robustness Diagram: Dispatcher answers

Use Case 3: Caller does not have an emergency

Dispatcher provides information or redirects call to automated voice response unit or another number, call is disconnected.

Figure 4.3: Robustness Diagram: Caller does not have an emergency

Use Case 4: Caller disconnects

Caller hangs up or call gets disconnected before all details of emergency are reported, dispatcher calls the caller, caller answers, dispatcher asks for a detailed account of accident.

Alternate Flow: Caller does not answer

Caller does not answer the phone.

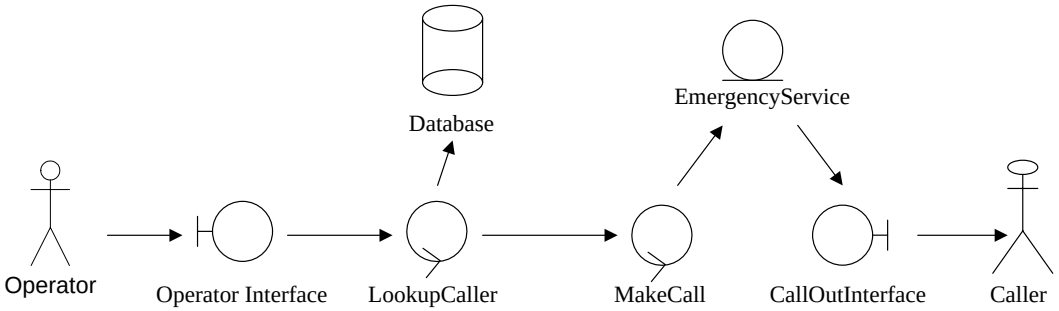

Figure 4.4: Robustness Diagram: Caller disconnects

Use Case 5: Hoax

Dispatcher retrieves phone number from system, if repeat offender, dispatcher sends information to police. Dispatcher disconnects line, dispatcher is free.

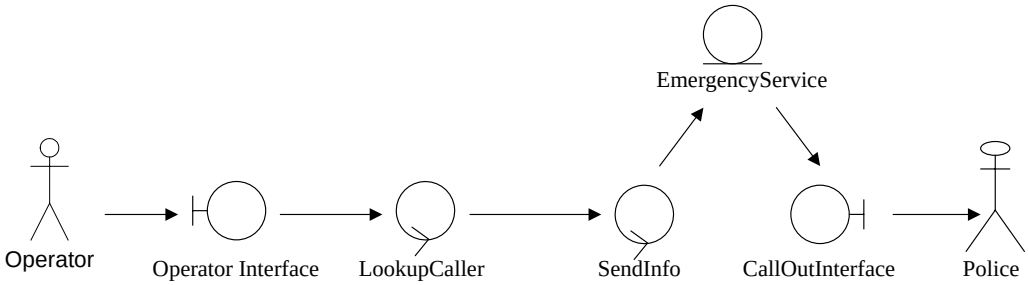

Figure 4.5: Robustness Diagram: Hoax

Use Case 6: Emergency

Dispatcher asks for a detailed account of incident, logs name and location of caller on screen, as well as any other info, and confirms location of caller on map. Dispatcher sets up conference call between caller and ambulance, if necessary. Dispatcher keeps caller on the line, all necessary emergency response personnel confirm arrival at scene. Call ends, dispatcher is free.

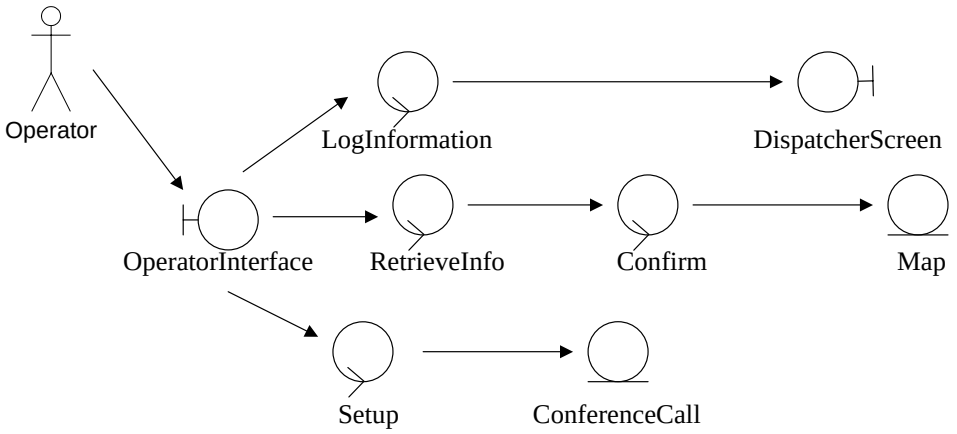

Figure 4.6: Robustness Diagram: Emergency

Use Case 7: Dispatcher sends info to ES personnel

Dispatcher sends accident information and map, if relevant, to ambulance, police and rescue personnel.

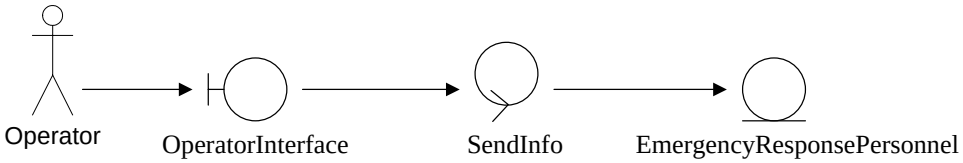

Figure 4.7: Robustness Diagram: Dispatcher sends info to ES personnel

Use Case 8: Dispatcher saves call info

Dispatcher logs the name of the caller, phone number, and the incident, as well as background information.

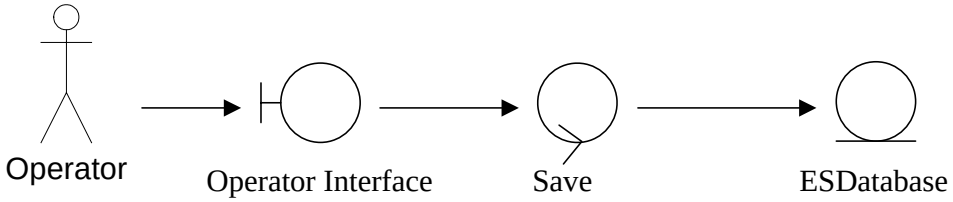

Figure 4.8: Robustness Diagram: Dispatcher saves call information

Use Case 9: Billing

Caller hangs up phone, billing information is debited from emergency service or govt. agency.

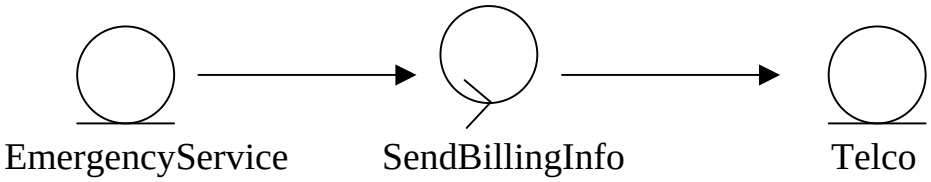

Figure 4.9: Robustness Diagram: Billing

Software design is not a linear process, with one step commencing as soon as the previous one has been finished. Each step is designed to take the design one step further and shows what has been left out of the previous step. For example, use case diagrams encourage refinement of the system by listing down exactly what the steps are that need to be taken. Robustness diagrams elaborate on use cases by making sure the use cases do not duplicate themselves, and that all the necessary objects have been included in the design. By the time robustness diagrams have been completed, the various classes of a system will be identified. Very often, in software design, one returns to a previous step and fine tunes its requirements and descriptions, perhaps even several times. For the sake of simplicity, all the reiterations of the various steps have been left out, and only the finished sections presented.

4.2 Message Sequence Diagrams

The next step after robustness diagrams is creating message sequence charts, which show the information processing of the application in more detail. These charts help refine the design further by visually showing methods called on the various interfaces that implement the emergency service. Although, strictly speaking, showing communication between interfaces belongs in the computational viewpoint, the message sequence charts resulting from the robustness diagrams in **Section 4.1** are too high level to be moved to the computational viewpoint, and as they mainly show the flows of information between the various classes, they are therefore placed here, in the information viewpoint.

The message sequence charts follow from the use cases and robustness diagrams, and should be self explanatory. Notes on the message sequence charts identify which use case is being illustrated.

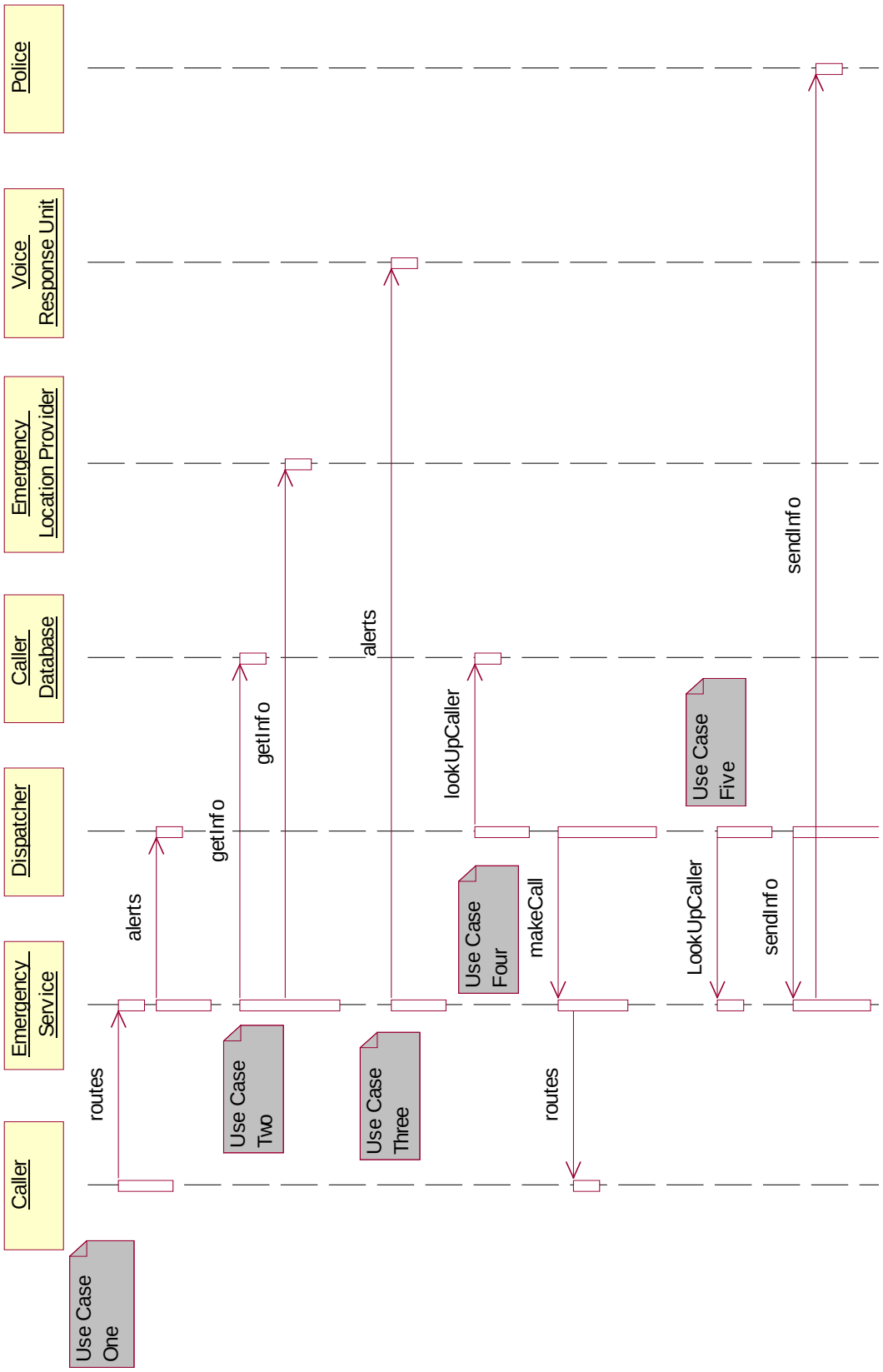

Figure 4.10: Message Sequence Chart: Use Cases 1 - 5

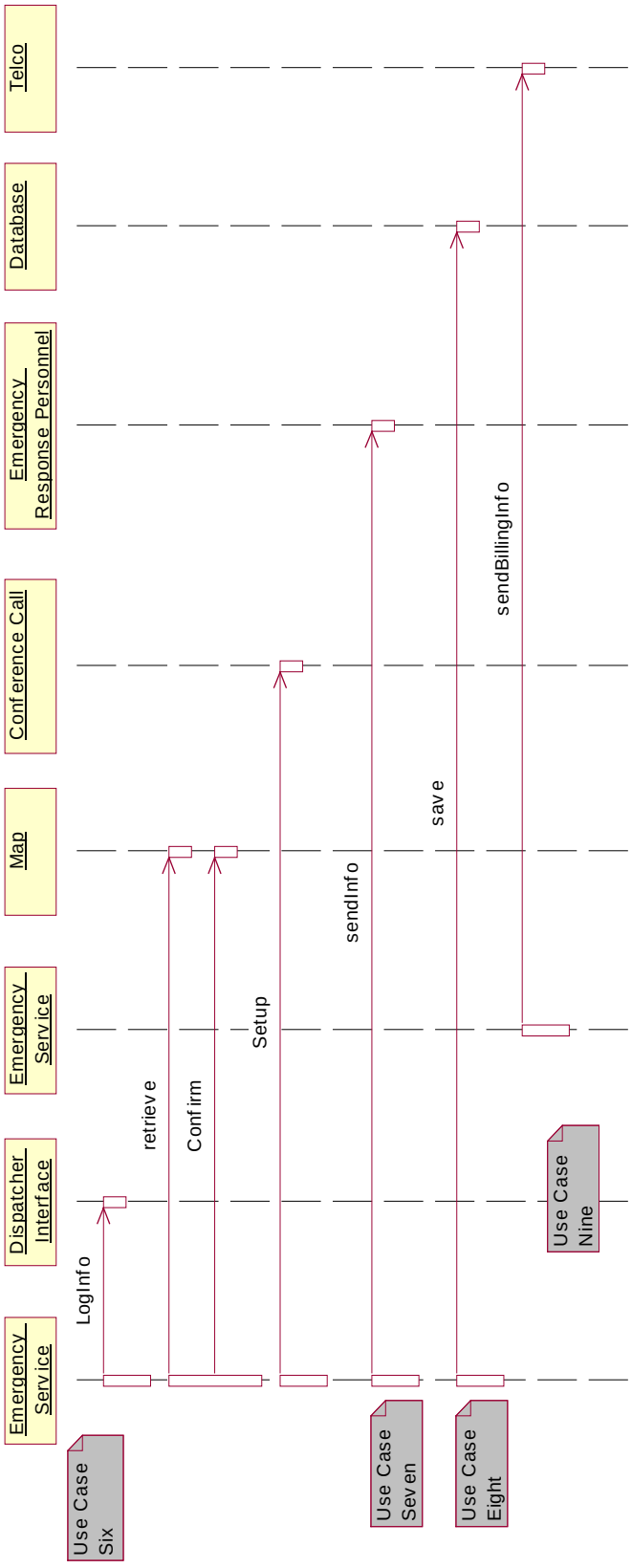

Figure 4.11: Message Sequence Chart: Use Cases 6 - 9

There is a fine line between the information and computational viewpoints. The message sequence charts at the end of this chapter were placed in the information viewpoint because they were not at a low enough level to identify the actual interfaces that are part of the service, and instead showed the information passed between the classes and objects.

Chapter 5

Computational Viewpoint

5.1 Introduction

The computational viewpoint of the RM-ODP is, at its simplest level, a snapshot of all the ways the components in the service interact with each other. It is a model of the system and its environment in terms of the individual, logical components which are sources and sinks of information. The viewpoint enables distribution through functional decomposition of the system into objects which interact at interfaces [28].

5.2 Computational Viewpoint of Emergency Service

Ideally, when one is designing a software program, according to design methodology, there are no constraints on the classes or methods of implementation. One works down through the design steps and levels almost in a vacuum, and then comes up with an implementation that can be converted to code. However, practically speaking, there are various platforms and implementations that will be used. These platforms already might have a certain structure, or classes that are used in the implementation, that will make the design process easier if used. This project is no exception.

As mentioned in **Section 2.7**, OSA/Parlay is being used to implement the service, and has information structures of its own which need to be used. For example, in the robustness diagram for the second use case, there is a class called EmergencyLocationProvider, that, as the name suggests, provides the location of a person calling the emergency service. This class maps to the OSA/Parlay Mobility SCF, which

does the same thing. In the same way, the entire functionality of the emergency service, apart from access to some databases, is encapsulated by the OSA/Parlay SCFs, which are reusable blocks of functionality. Because of this, in the computational viewpoint, we are somewhat constrained in the design of this level of the service, because we have the constraints of the OSA/Parlay objects we are using.

The relation between the objects in the gateway has already been defined in the OSA/Parlay standards, so the only thing left to define is the relationship between the databases, the database manager, and the application. For completeness, **Figure 5.1**, which is the component diagram, shows a complete overview of all the components within the service and how they communicate with each other.

As mentioned above and seen in the component diagram of the emergency application, apart from the databases and database manager, the rest of the functionality of the emergency service lies in the SCFs within the OSA/Parlay gateway. **Figure 5.1** visually illustrates how the SCFs that are used in this service make up the OSA/Parlay gateway, as well as illustrating the open distributed architecture of the emergency application, with the emergency application logically falling in the application layer, the OSA/Parlay gateway and the functionality it contains belonging to the service control layer, and below that, the network resource layer.

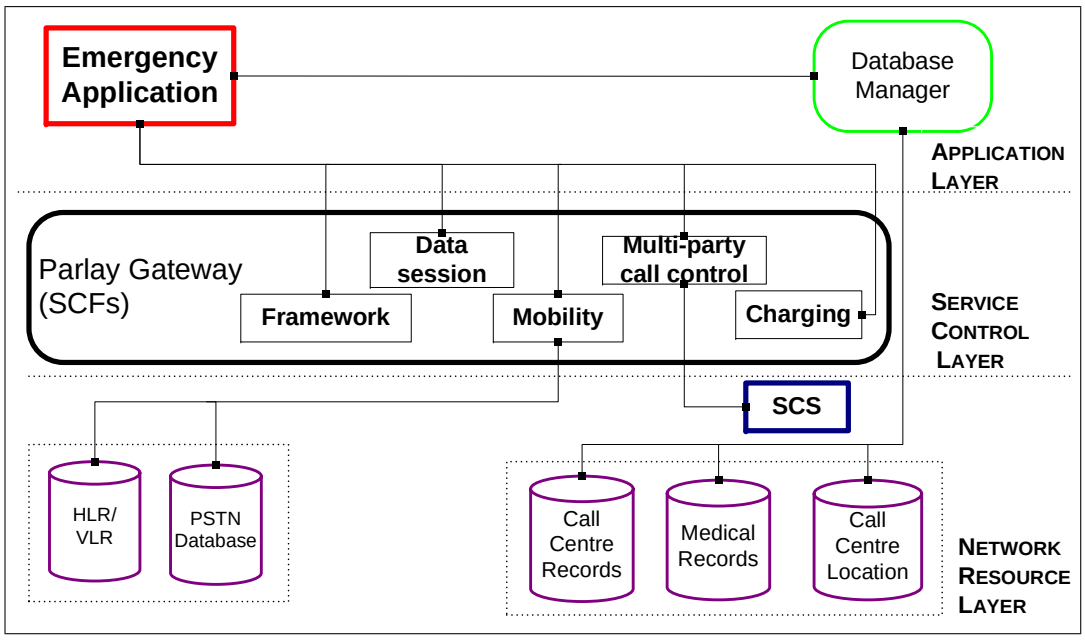

Figure 5.1: Computational Diagram of the Emergency Service

5.2.1 OSA/Parlay Information Structures

From now on in the design process, we will be largely relying on pre-defined objects that are part of the Parlay implementation. These pre-defined objects will be combined with the classes and objects identified in the previous chapter, and will be illustrated by message sequence charts. As these message sequence charts, which are displayed below, decompose the application into the OSA/Parlay SCF interfaces and illustrate how the interfaces communicate with each other, they find their place in the computational viewpoint [28].

The **Authentication**, **Service Discovery** and **Service Selection** message sequence charts are part of the initialization and authentication of the application, and only occur the first time the application is online. As such, these are not part of the design of the service, and are not represented in the use cases.

The **Call Initiation**, **Number Translation and Location** and **Call Connection**message sequence charts are represented by the first and second use cases, where the caller dials the emergency number, gets routed to the closest emergency centre, the dispatcher answers, and obtains the location of the caller.

As all call centres have infrastructures to assist them in managing and answering calls, some of the functionality that was originally put in the use cases, like the third use case, which connects a caller who does not have an emergency, but needs information, to a automatic voice response unit, is not represented by a OSA/Parlay message sequence chart. It is assumed that every call centre will have this application as part of its infrastructure, and to include it in the design of the emergency service will be repetition of functionality.

The **Call Disconnect** and **Call Connection** message sequence charts represent the fourth use case, where the caller disconnects and the dispatcher in the emergency centre calls the original caller.

The emergency application will use a database manager, as illustrated in **Figure 5.1**, to retrieve information from the databases, such as a hoax caller's phone number. The **Data Connection with Emergency Service Provider** message sequence chart then represents the dispatcher sending the hoax caller's information to the police.

The sixth use case, in which the dispatcher deals with an emergency, is represented by the **Call Connect** message sequence chart, in which the dispatcher simply adds

another leg to the call, to communicate in a conference call with the emergency response personnel, and uses the database manager to retrieve and save information.

The **Data Connection with Emergency Service Provider** message sequence chart represents the seventh use case, in which the dispatcher sends information about the emergency to ambulance, police or other rescue personnel.

The eighth use case will be executed by the database manager, which will save information about the emergency to the relevant databases.

And finally, the **Reverse Billing** message sequence chart represents the ninth use case, in which the telco or government agent responsible for the emergency centre is billed for the call.

5.3 Message Sequence Charts

- The **Authentication** message sequence chart was copied exactly from the **Framework SCF** standards [30].
- The **Service Discovery** message sequence chart was adapted from the Service Subscription section of the **Framework SCF** standards [31] by eliminating the describeServiceType() and discoverService() methods. The services needed for the emergency service are generic and common enough to eliminate the need for finding out their properties, which is what these two methods do.
- The **Service Selection** message sequence chart was adapted from the Service Selection and the Sign Service Agreement sections of the **Framework SCF** standards [32] by combining the two.
- The **Call Initiation** message sequence chart was copied from the Call Forward on Busy service in the **Multi-Party Call Control SCF** [33], with only a getInfoReq() method inserted to request information associated with the call leg to party A to calculate charging.
- The **Number Translation** message sequence chart incorporates the Subscription and Network Induced Location Report message sequence chart in the **Mobility SCF**, designed specifically for emergency calls, which states that "The following sequence diagram shows how an application subscribes to emergency location reports from the Emergency User Location service. When the User Location Emergency service receives Network Induced Location requests (triggered by emergency calls) it reports that to the application" [34].

- The **Call Connection** message sequence chart is very similar to the continuation of the Call Forward on Busy service in the **Multi-Party Call Control SCF** [33], with the methods relating specifically to the call forward on busy service being removed, and a timer inserted so that if the call is not routed to a call centre in a specific amount of time, the call will be routed to the next closest emergency call centre.

- The **Call Disconnect** message sequence chart is copied from the last third of the Call Information Collect service in the **Multi-Party Call Control SCF** [35].

- The **Data Connection with Emergency Service Provider** message sequence chart, which is designed to, for example, send the details of a break-in to the police in a text file, has been adapted from the Address Translation with Charging section in the **Data Session Control SCF** [36].

- And lastly, the **Reverse Billing** message sequence chart is taken from the Immediate Charge section in the **Charging SCF** [37]

5.3.1 Authentication

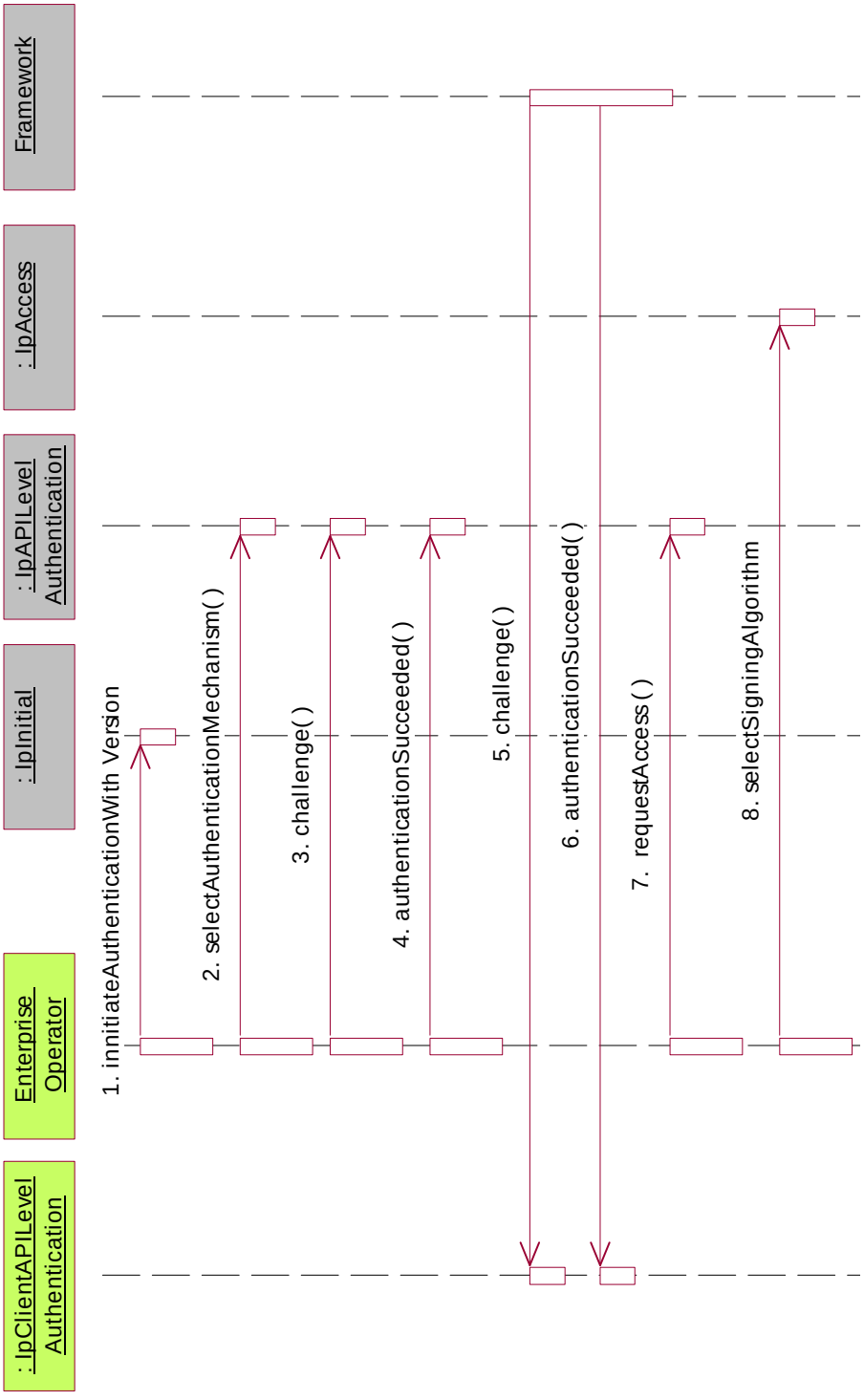

Figure 5.2: Sequence Diagram: Authentication

I. Authentication

1. *initiateAuthenticationWithVersion* - The client invokes this method on the Framework's public (initial contact) interface to initiate the authentication process. It provides in turn a reference to its own authentication interface. The Framework returns a reference to its authentication interface.

2. *selectAuthenticationMechanism* - The client invokes this method on the Framework's API Level Authentication interface, identifying the authentication algorithm it supports. The Framework prescribes the method to be supported.

3. *challenge* - The client authenticates the Framework, issuing a challenge with this method.

4. *authenticationSucceeded* - The client provides an indication if authentication succeeded.

5. *challenge* - The Framework authenticates the client.

6. *authenticationSucceeded* - The Framework provides an indication if authentication succeeded.

7. *requestAccess* - Upon successful authentication of the client by the Framework, the client is permitted to invoke this method on the Framework's API Level Authentication interface, providing in turn a reference to its own access interface. The Framework returns a reference to its access interface.

8. *selectSigningAlgorithm* - The client and framework negotiate the signing algorithm to be used for any signed exchanges.

5.3.2 Service Discovery

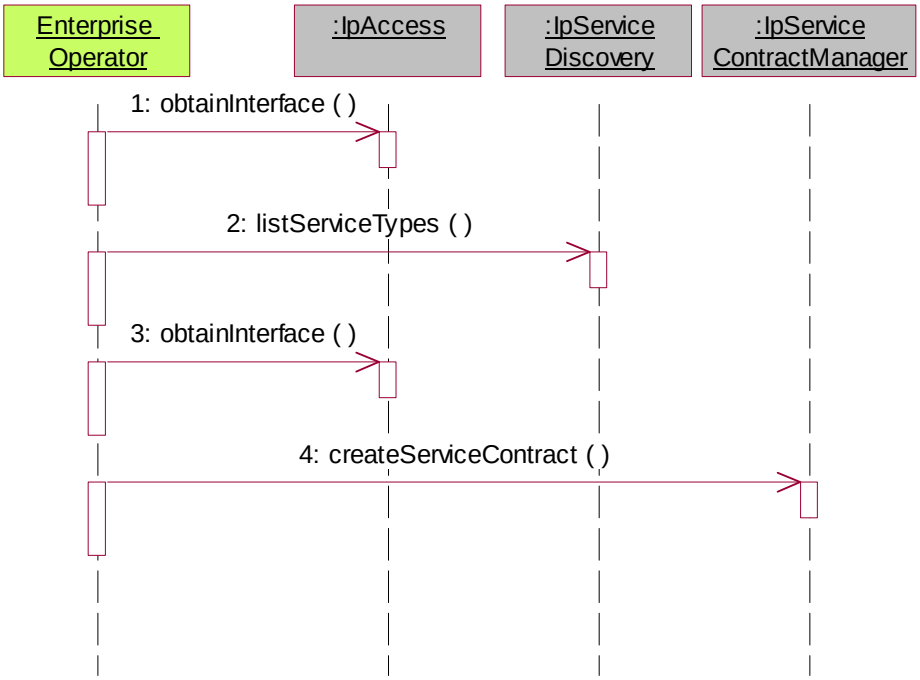

Figure 5.3: Sequence Diagram: Service Discovery

II. Service Discovery

1. *obtainInterface* - The client invokes this method on the Framework's Access interface to obtain a reference to its service discovery interface.
2. *listServiceTypes* - The application asks the Framework what service types are available from this network, learning what SCF's are supported by the network.
3. *obtainInterface* - The client invokes this method on the Framework's Access interface to obtain a reference to its service contract manager interface.
4. *createServiceContract* - This method is used to subscribe to a service.

5.3.3 Service Selection

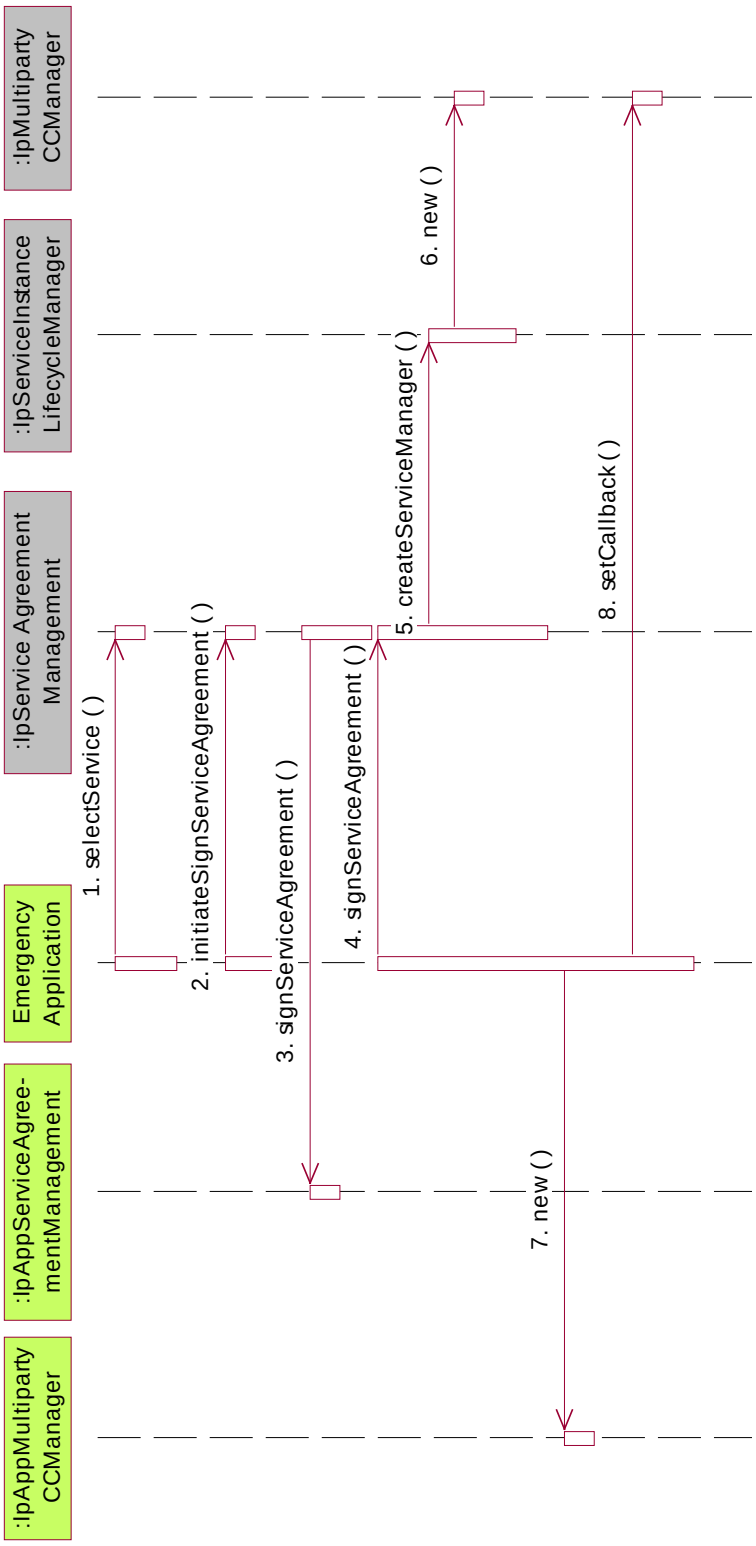

Figure 5.4: Sequence Diagram: Service Selection

III. Service Selection

1. *selectService* - The application selects the service, using a service ID. A Service Token is returned to the application.
2. *initiateSignServiceAgreement* - The client application initiates the process of signing the service agreement.
3. *signServiceAgreement* - The client application signs the service agreement
4. *signServiceAgreement* - The framework signs the service agreement. As a result, a service manager interface is returned to the application.
5. *createServiceManager* - The framework requests the service identified by the service ID to return a service manager interface reference.
6. *new* - The lifecycle manager creates a new call control manager for the application.
7. *new* - The application creates a new call control manager interface to be used for callbacks.
8. *setCallback* - The application sets the callback interface to the interface created with the previous message.

5.3.4 Call Initiation

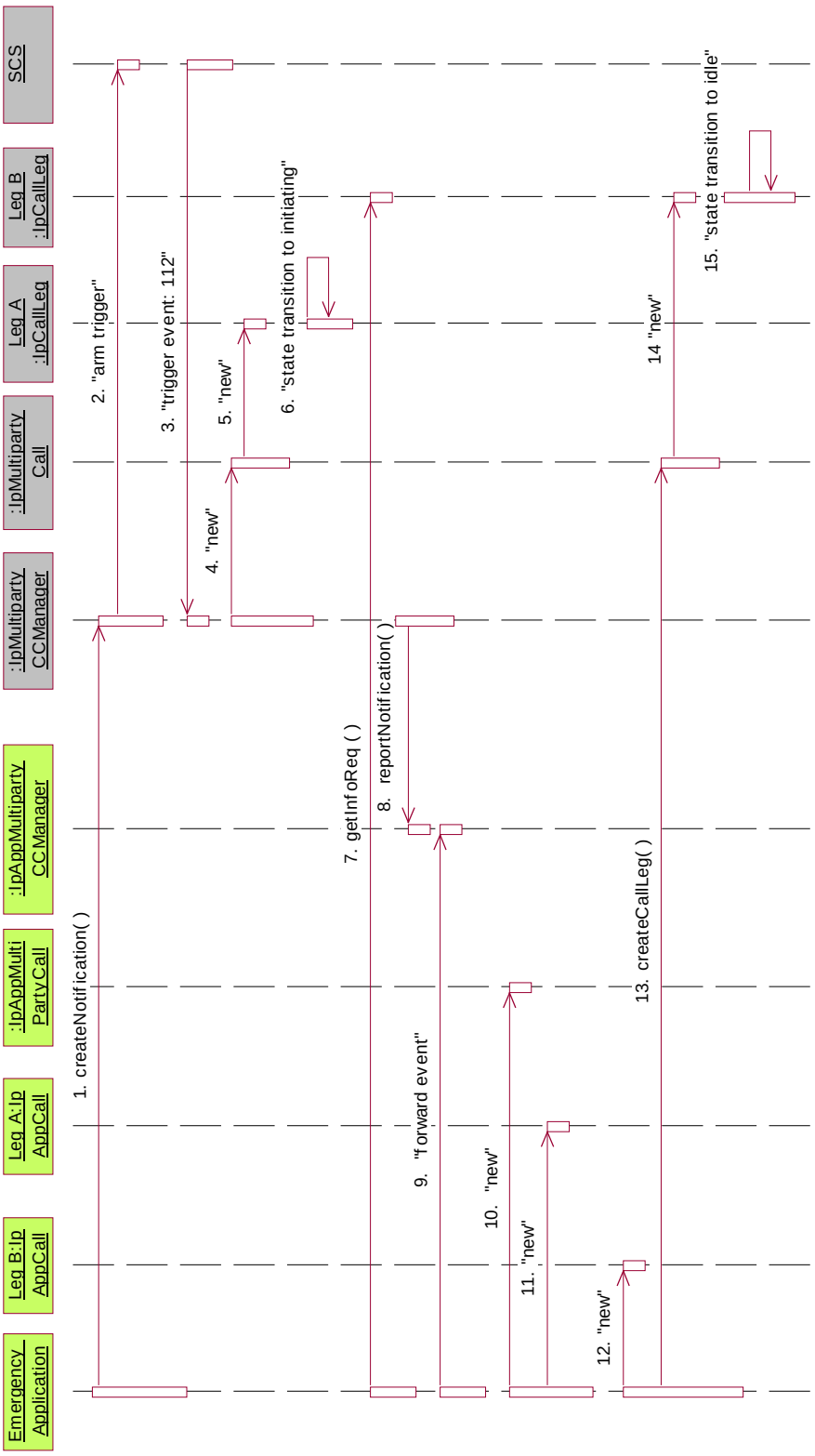

Figure 5.5: Sequence Diagram: Call Initiation

IV. Call Initiation

1. *createNotification* - This message is sent by the application to enable notifications on new call events.
2. *arm trigger* - A trigger is set, asking the SCS to inform the call control manager of calls matching the desired specification.
3. *trigger event* - When a new call, that matches the event criteria, arrives a message is sent to the call control manager.
4. *new* - A new multi party call object is created to handle this specific call.
5. *new* - A new call leg is created, corresponding to the calling party.
6. *state transition to initiating* - The new call leg instance transits to state initiating.
7. *getInfoReq* - The application requests information associated with the call leg to party A to calculate charging.
8. *reportNotification* - This message is used to pass the new call event to the object implementing the IpAppMultiPartyCallControlManager interface.
9. *forward event* - This message forwards the previous message to the application logic.
10. *new* - The application uses this message to create an object implementing the application multi party call interface. The reference to this object is passed back to the object implementing the IpMultiPartyCallControlManager using the return parameter of the reportNotification.
11. *new* - A new application call leg is created to receive callbacks for the leg corresponding to party A.
12. *new* - A new application call leg is created to receive callbacks for the second leg of the call.
13. *createCallLeg* - This message is used to create a new call leg object. The object is created in the idle state and not yet routed in the network.
14. *new* - A new call leg corresponding to party B is created.
15. *state transition to idle* - A transition to idle state is made after call leg B has been created.

5.3.5 Number Translation and Location

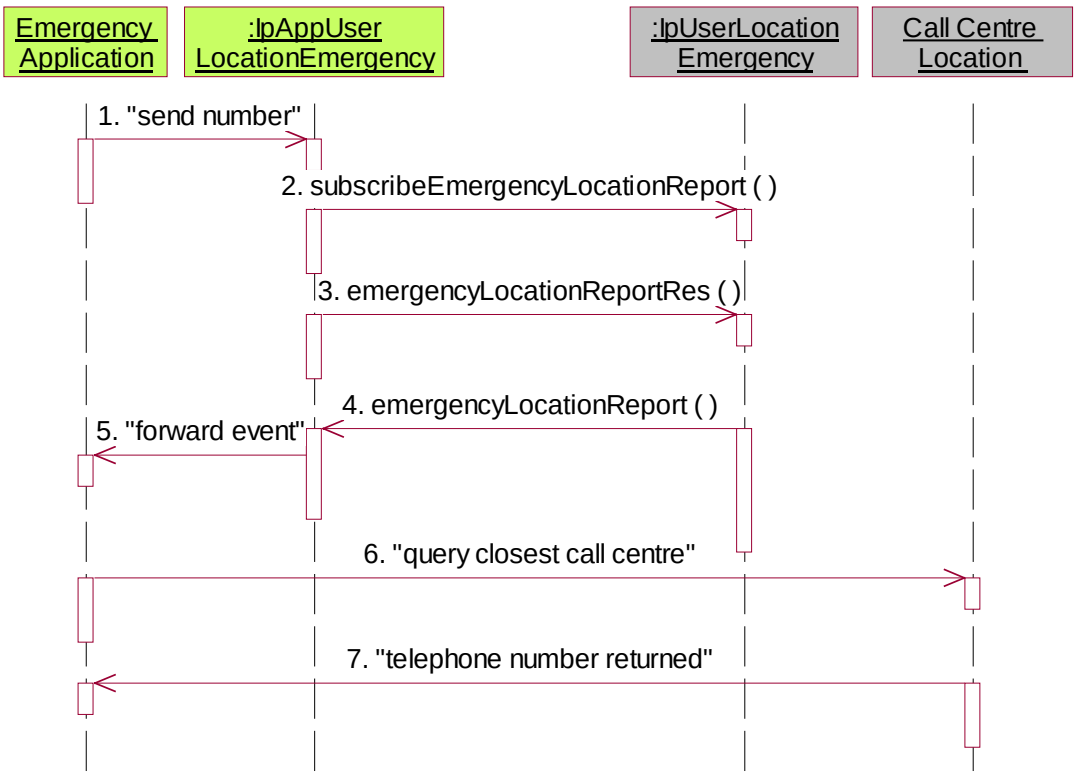

Figure 5.6: Sequence Diagram: Number Translation

V. Number Translation and Location

1. *send number* - The application logic sends the mobile number to the application UserLocationEmergency interface.

2. *subscribeEmergencyLocationReport* - The application subscribes to emergency location reports. This method has been included for the sake, and will only be used once, the very first time the service is online.

3. *emergencyLocationReportReq* - The application requests a report on the location of a user making an emergency call. This method also specifies the identity of the user or terminal, the requested location type, accuracy, response time, and most importantly, the priority of the response.

4. *emergencyLocationReport* - When the mobility service receives a network induced location request, this method passes the location and other requested information on the user to its callback object.

5. *forward event* - The location of the caller is forwarded to the application

6. *query closest call centre* - The application queries the emergency call centre location database to see which call centre is closest to the caller.

7. *telephone number returned* - The database returns the telephone number of the closest emergency call centre.

8. ** NOTE - there is also the possibility of including a triggered location report for mobile users. This method will send a location report to the application every time the user's location changes. However, for the sake of simplicity, this method is not included.*

5.3.6 Call Connection

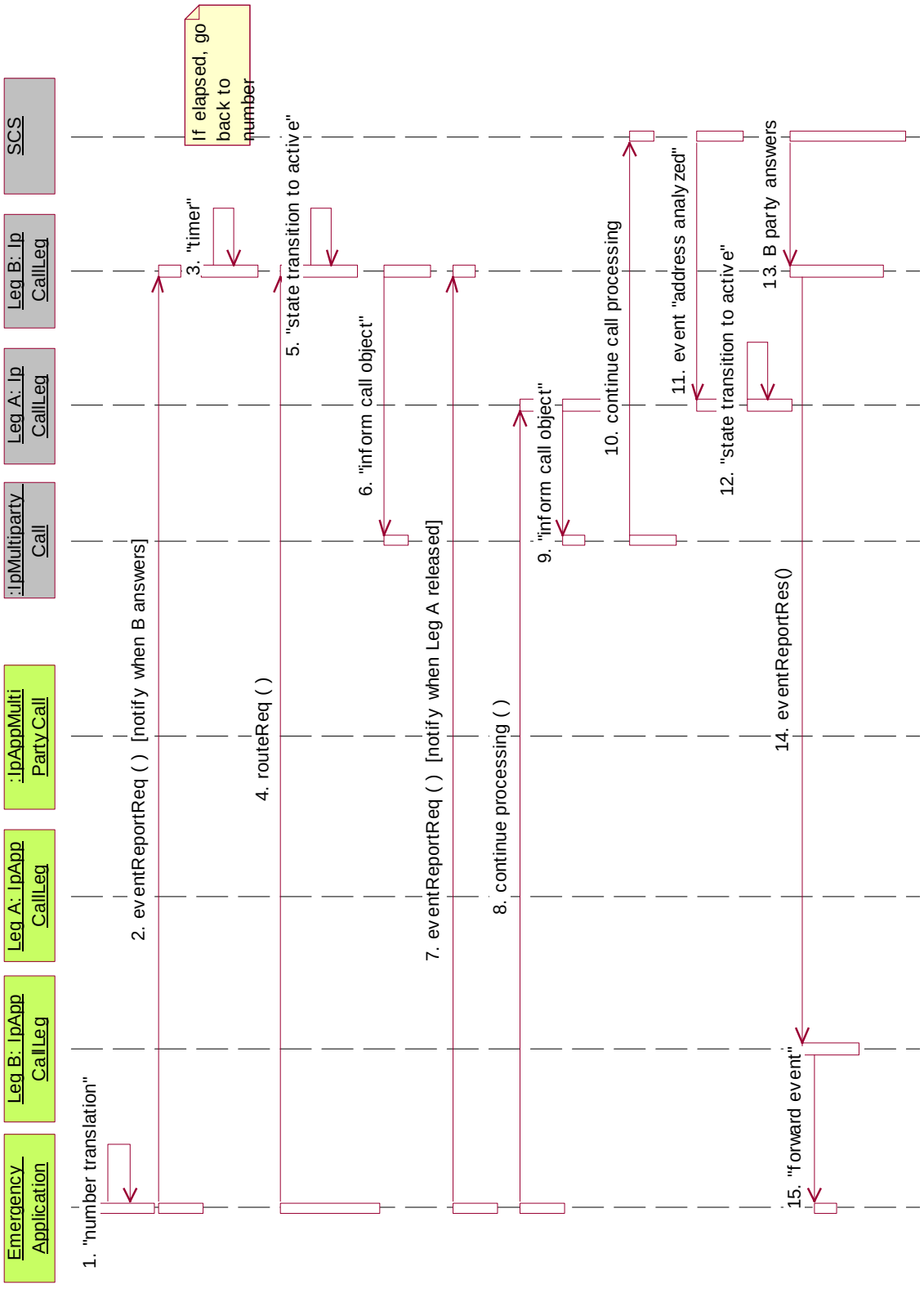

Figure 5.7: Sequence Diagram: Call Connection

VI. Call Connection

1. *number translation* - This refers to the process by which the dialed number "112" is translated to the number of the closest emergency call centre, and is covered in detail in the seventh message sequence chart.

2. *eventReportReq* - The application requests to be notified when the leg to party B answers.

3. *timer* - A timer is set, and if the call is not answered in the specified amount of time, the call is routed to the next closest emergency call centre.

4. *routeReq* - The application requests to route leg B to the closest emergency call centre.

5. *state transition to active* - The call leg B instance transits to active state.

6. *inform call object* - The multiparty call object is informed of the state of call leg B.

7. *eventReportReq* - The application requests to be notified when the leg to the A party is released.

8. *continue processing* - The application requests to resume call processing for the originating call leg.

9. *inform call object* - The multiparty call object is informed of the resumption of call processing.

10. *continue call processing* - The multiparty call object requests the network to resume call processing to reach the associated party as specified by the application.

11. *event "address analyzed"* - The originating call leg is notified that the number provided by the application has been analyzed by the network.

12. *state transition to active* - The originating call leg makes a transition to active.

13. *B party answers* - When the B party answers the call, the termination call leg is notified.

14. *eventReportRes* - The object implementing party B's IpCallLeg interface passes the result of the call being answered back to its callback object.

15. *forward event* - This answer message is then forwarded to the application logic.

5.3.7 Call Disconnect

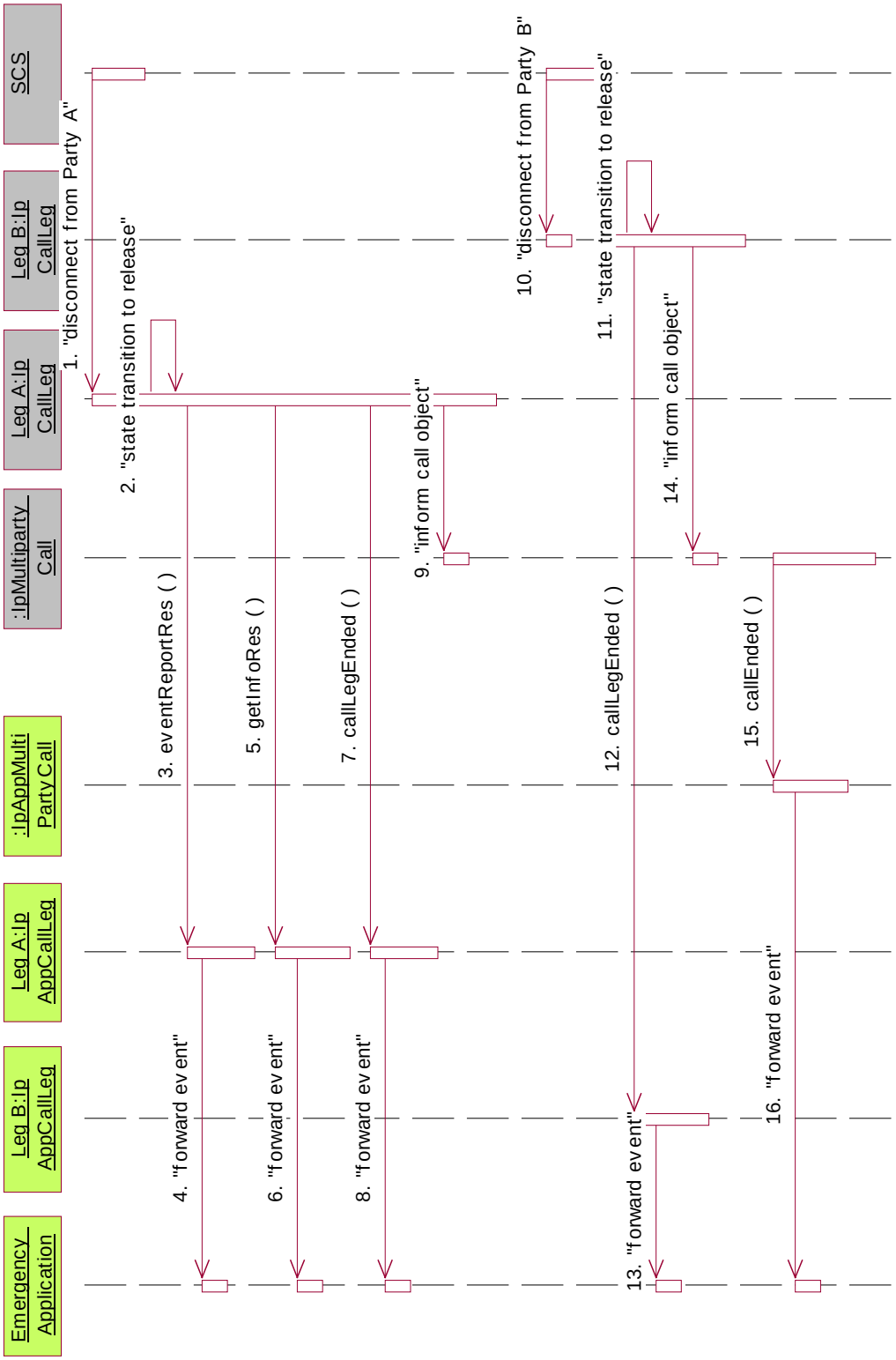

Figure 5.8: Sequence Diagram: Call Disconnect

VII. Call Disconnect

1. *disconnect from A party* - The termination call leg is notified when party A hangs up.
2. *state transition to releasing* - A transition to released state is made after the call leg has been disconnected.
3. *eventReportRes* - The object implementing party A's IpCallLeg interface passes the result of the call being answered back to its callback object.
4. *forward event* - This answer message is then forwarded to the application logic.
5. *getInfoRes* - The call leg information is reported.
6. *forward event* - The event is forwarded to the application logic.
7. *callLegEnded* - The initiating call leg is destroyed, and AppLeg A is notified.
8. *forward event* - This answer message is forwarded to the application logic.
9. *inform call object* - The multiparty call object is informed of the termination of Leg A.
10. *callLegEnded* - The terminating call leg is destroyed, and AppLeg B is notified.
11. *forward event* - This event is forwarded to the application logic.
12. *inform call object* - The multiparty call object is informed of the termination of Leg B.
13. *callEnded* - When all legs have been destroyed, the IpAppMultiPartyCall interface is notified that the call is ended.
14. *forward event* - The event is forwarded to the application logic.

5.3.8 Data Connection with Emergency Service Provider

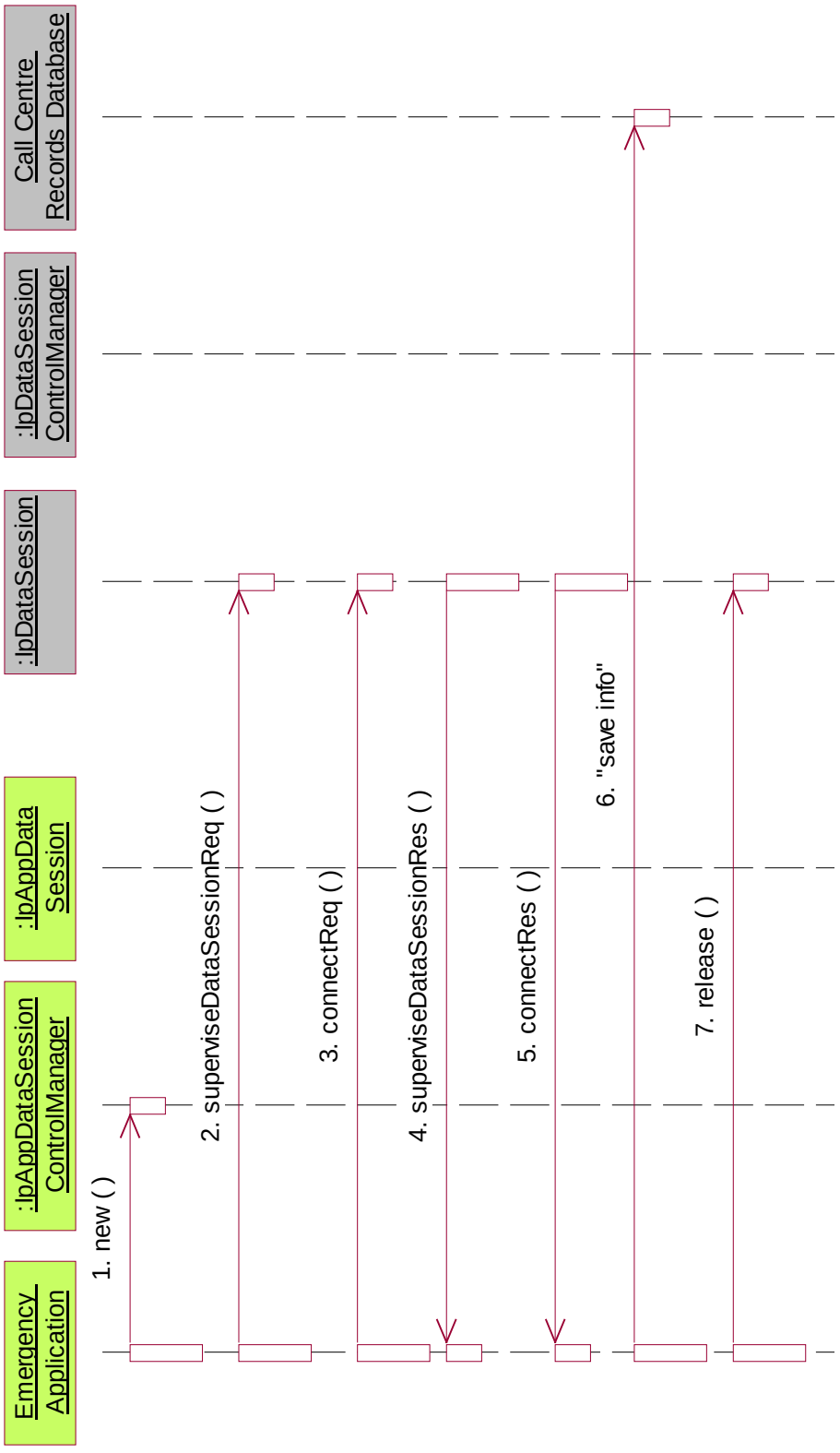

Figure 5.9: Sequence Diagram: Data Connection with Emergency Service Provider

VIII. Data Connection with Emergency Service Provider

1. *new* - The application is started, and creates a new IpAppDataSessionControlManager interface to handle callbacks.

2. *superviseDataSessionReq* - The application calls this method to supervise a data session.

3. *connectReq* - The application requests the connection of a data session with the destination party.

4. *superviseDataSessionRes* - The Data Session object reports a data session supervision event to the application.

5. *connectRes* - The Data Session object indicates that the request to connect a data session with the destination party was successful.

6. *save info* - The application saves the information about the call to the Emergency database.

7. *release* - The application requests the release of the data session and associated objects.

5.3.9 Reverse Billing

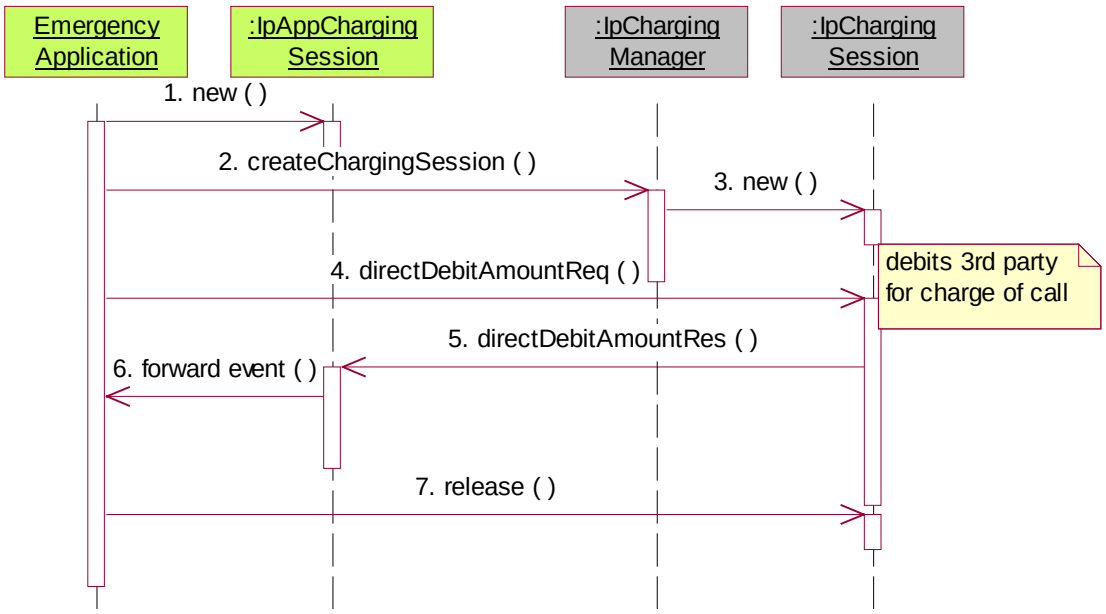

Figure 5.10: Sequence Diagram: Reverse Billing

IX. Reverse Billing

1. *new* - The application creates a local object implementing the IpAppCharging Session interface.
2. *createChargingSession* - The application opens a charging session.
3. *new* - An object implementing the IpChargingSession interface is created
4. *directDebitAmountReq* - A request is sent for the third party to be debited for the charge of the call.
5. *directDebitAmountRes* - The charging session indicates that the request was successful.
6. *forward event* - The message is forwarded to the application logic.
7. *release* - The application frees all resources associated with the charging session.

The following sections take a closer look at each of the SCFs used by the emergency application, and the interfaces used by each SCF.

5.3.10 Computational View of Emergency Application

The computational view of the Emergency Application is shown in **Figure 5.11**. This figure illustrates all the callback interfaces the application uses to communicate with the various interfaces of the SCFs in the Parlay Gateway.

The **IpClientAPILevelAuthentication** callback interface is used by the Framework to authenticate the application. The Framework challenges the application, which must respond with the correct response. When the framework has authenticated the application, it sends an authenticationSucceeded method to the **IpClientAPILevelAuthentication**callback interface [27].

After the application has used the **IpServiceAgreementManagement** interface to identify a service that it wants to use, that framework interface then invokes the application **IpAppServiceAgreementManagement** callback interface to initiate signing a service agreement [27].

The Multi-party Call Control interface sends a message to it's callback interface, **IpMultipartyCallControlManager**, to inform it about the new call event. The callback interface then forwards that information to the emergency application logic [38].

The **IpAppMultipartyCall** callback interface gets information from either the Multi-party Call Control manager or the emergency application about when a new call is initiated or ended, depending on which side it is initiated, and passes the information to its callback interface or the emergency application [38].

The call leg callback interfaces, **IpAppCallLeg**, are informed by the initiating side, either the application logic or the Multi-party Call Control manager, when there has been a change of status in a call leg, or when there needs to be a change of status [38].

In the case of an emergency call, the network may locate the caller automatically by sending a request to the **IpUserLocationEmergency** interface. The resulting location is then sent to the application using a callback method for the **IpAppUserLocationEmergency** interface [17].

The **IpAppDataSessionControlManager** callback interface is created by the emergency application to manage the data session [39].

The **IpAppDataSession** callback interface provides basic methods for applications

to control data sessions [39].

The **IpAppChargingSession** callback interface is sent a method by the **IpChargingSession** interface, informing it that the debit has gone through. The callback interface then forwards that message on to the emergency application [40].

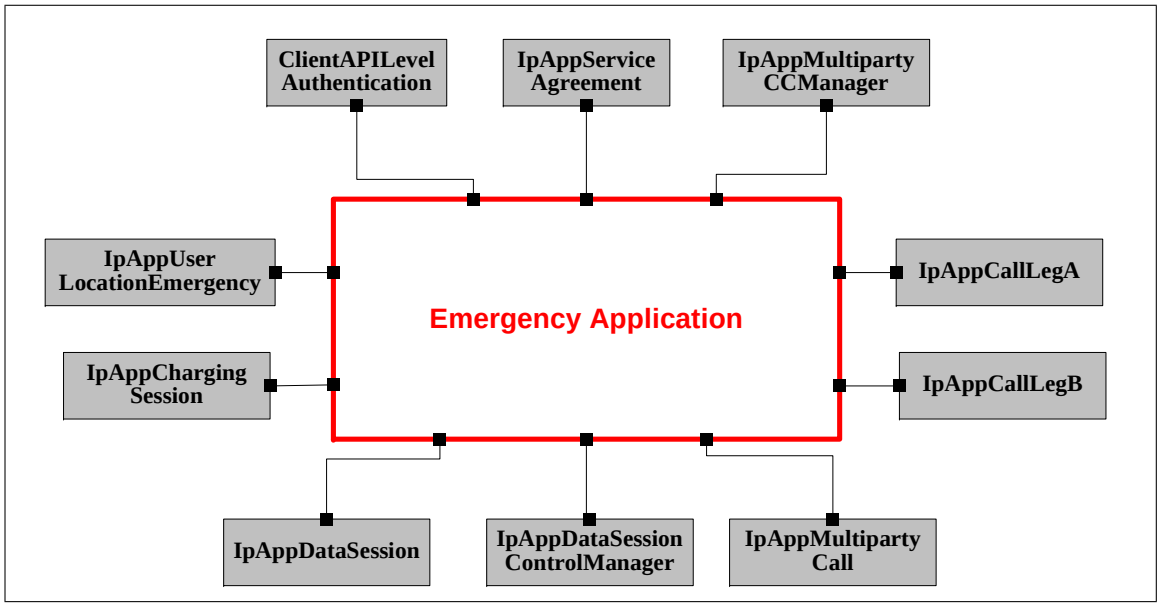

Figure 5.11: Computational Diagram of the Interfaces used in the Emergency Application

5.3.11 Computational View of Framework SCF

Figure 5.12 shows the interfaces used by the emergency application in the Framework SCF.

The **IpIntial** Initial Framework interface is used by the application to initiate the authentication process with the Framework. The interface is implemented by the Framework, and implements the initiateAuthenticationWithVersion() method [27].

The **APILevelAuthentication** Framework interface is also used by the application to authenticate the Framework, and it initiates the actual authentication process. It selects the authentication mechanism and authenticates itself for the application. After authentication of both the application and the Framework have been done, the application invokes the requestAccess( ) method to get a reference to the Framework's access interface [27].

As the name suggests, **IpAccess** Framework interface is used by the application to facilitate access to the Framework by obtaining a reference to its service discovery interface [27].

After the **IpAccess** interface passes the application a reference to its service discovery interface, **IpServiceDiscovery**, the application queries it as to what types of services are supported by the Framework. The interface returns a list of all service types currently supported by the Framework [27].

After the application has chosen the services it needs, it creates a service contract on the **IpServiceContractManagement** interface [27].

The application then uses the **IpServiceAgreementManagement** interface to identify a service that it wants to use. The **IpServiceAgreementManagement** interface is then invoked by the application to initiate signing a service agreement, and then invokes the signServiceAgreement method on the application's service agreement management interface. After the service agreement has been signed by both the application and the Framework, the **IpServiceAgreementManagement** interface requests the identified service to return a service manager interface reference [27].

Each service has a service manager interface that is the initial point of contact for the service. The **IpServiceInstanceLifecycleManager**interface allows the framework to get access to this service manager interface, and in this instance, creates a new call control manager for the application [27].

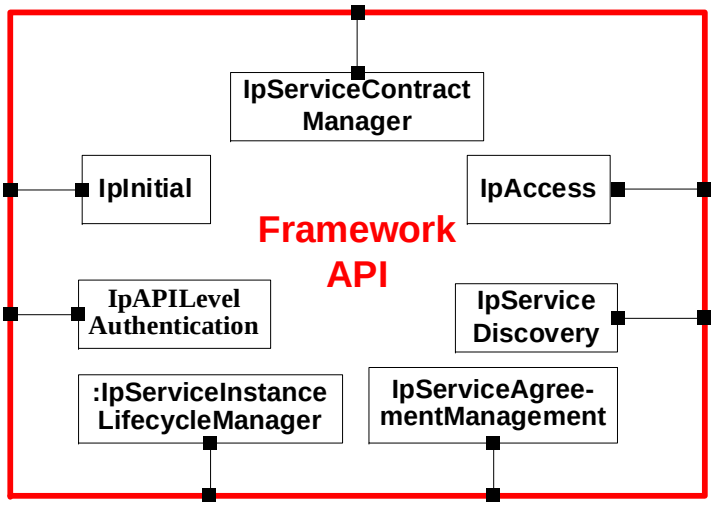

Figure 5.12: Computational Diagram of the Interfaces used in the Framework API

5.3.12 Computational View of Multiparty SCF

The next diagram, illustrated in **Figure 5.13** shows the interfaces the Multiparty Call Control SCF uses in the emergency application.

The **IpServiceInstanceLifecycleManager** interface in the Framework SCF creates the **IpMultipartyCallControlManager**, which arms a trigger with the SCS so that the Call Control Manager is informed when a call is made to the emergency service. The Multi-party Call Control service covers the functionality of the Generic Call Control service, with the addition of leg management, which allow for multy-party calls to be established. The Multi-party Call Control Service is represented by the IpMultiPartyCallControlManager interface. The **IpMultipartyCallControlManager** interface creates an **IpMultipartyCall** interface, that represents the call being made from the caller to the emergency service, as well as reporting to its callback interface [38].

The **IpMultipartyCall** interface handles the creation of new call legs for the current call, getting requests both from the Multi-party Call Control Manager, as well as the emergency service. It receives information about the states of the call legs, as well as when the call ends [38].

Logically, there are as many call leg interfaces as there are call legs, since each interface represents a single call leg. The call leg interfaces, **IpCallLeg**, are informed by the relevant interface, either from the Multi-party Call Control SCF side or the application side when something needs to be done to a call leg, be it creating a new one, or disconnecting it. The call legs are also informed when an event has happened to a call leg, like a caller hanging up [38].

5.3.13 Computational View of Mobility SCF

Figure 5.14 shows the interfaces the Mobility SCF uses in the emergency application for finding the location of the caller.

In the case of an emergency call, the network may locate the caller automatically by sending a request to the **IpUserLocationEmergency** interface. The resulting location is then sent to the application using a callback method for the **IpAppUserLocationEmergency** interface [17].

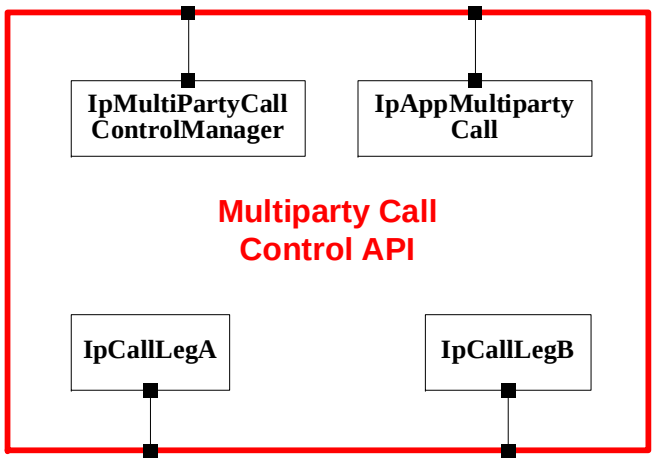

Figure 5.13: Computational Diagram of the Interfaces used in the Multi-party Call Control API

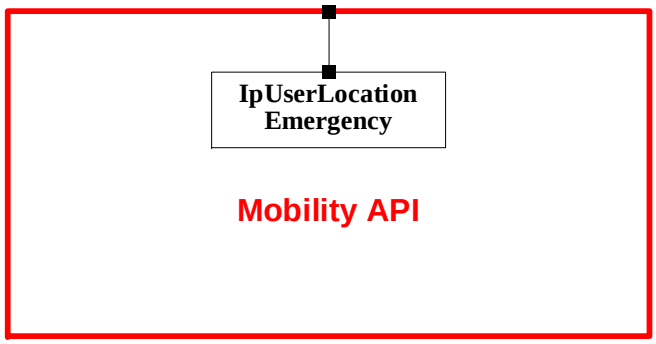

Figure 5.14: Computational Diagram of the Interfaces used in the Mobility API

5.3.14 Computational View of Data Session Control SCF

Figure 5.15 shows the interfaces the Data Session Control SCF uses in the emergency application.

The **IpDataSessionControlManager** interface, as the name suggests, manages a data session by supervising the connection and release of the session [39].

The **IpDataSession** interface provides basic methods for applications to control data sessions [39].

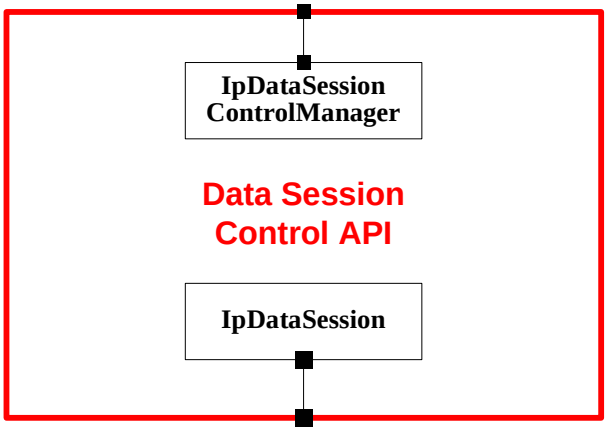

Figure 5.15: Computational Diagram of the Interfaces used in the Data Session Control API

5.3.15 Computational View of Charging SCF

Figure 5.16 shows the interfaces the Charging SCF uses in the emergency application.

The **IpChargingManager** interface manages the charging service by creating a new charging session and providing management functions to the charging service [40].

The **IpChargingSession** interface actually controls the charging receiving a method informing it of who and how much to debit or charge, and implementing that [40].

5.4 Engineering Viewpoint

The computational viewpoint, described in the previous section, decomposed the emergency system functionality into a set of components that interact with each other through interfaces. The component interfaces communicate as if they are part of a single processing system, with their distributed nature being hidden or abstracted. Therefore, mechanisms are needed to provide distribution functionality for these component interfaces, to enable transparency.

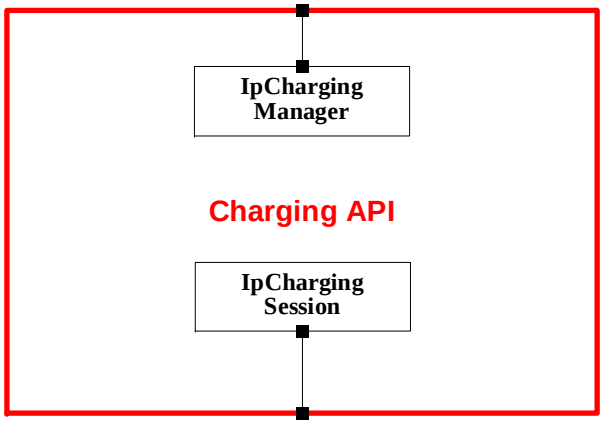

Figure 5.16: Computational Diagram of the Interfaces used in the Charging API

The engineering viewpoint, discussed briefly in this section, describes the supporting components used to enable distribution transparency and the implementation of these components by focusing on the mechanisms and functions required to support distributed interaction between objects in the system. In other words, the engineering viewpoint of the RM-ODP takes the components described in the computational viewpoint and defines a networked computing infrastructure that supports distribution transparency for these components" [28].

Different architectures and implementations are available for creating distribution transparency of components, with the most commonly used standard being the Common Object Request Broker Architecture (CORBA) [41].

5.4.1 Introduction to CORBA

The current computing scene consists of numerous networks, each using different hardware architectures and operating systems, and hosting applications implemented in various languages. These networks, in isolation from each other, work perfectly well, but when it comes to interworking between them, problems of incompatibility arise. At the same time, object-oriented programming, which allows software code to be reused, is growing in popularity. An ideal situation would be for programmers to be able to write distributed applications across heterogenous platforms, quickly and easily, and reuse code and applications written in different languages.

As mentioned above, many architectures and frameworks have been devised to address the challenges of distributed computing. Perhaps the most thorough and accepted of the proposals is the Object Management Group's [42] contribution in the form of CORBA, which is an open, vendor independent architecture that enables computer applications to work together over any network, irrespective of the type of computer, operating system, programming language or network.

CORBA is not a certain type of implementation, but a set of standards that enable application developers to focus their efforts on creating applications instead of being concerned with common programming and distributing concerns. These standards have to be adhered to by anyone wishing to develop a CORBA-compliant application.

The CORBA standard defines an architecture of concepts and components that provide support to applications that are distributed over a heterogenous network.

In the light of the high standards with which CORBA has been devised, it is recommended to use CORBA for the distributed nature of the emergency service.

5.5 Technology Viewpoint

The Technology viewpoint focuses on the choice of technology for implementing a system, and describes how it is structured in terms of hardware and software components [28]. It provides a link between the design in the viewpoints and the actual implementation of the service.

As the main focus of this project is in the design of a service, the technology viewpoint is not specified, since the emergency service has not been implemented.

Chapter 6

Conclusion

6.1 Summary

This report set out to illustrate the principles of designing an application by a third party provider, which could then be offered to any telco, irrespective of the functionality used on their platform, and to design a full information structure for an emergency call centre service, which can be offered as a service or application on any core network, working towards the design of a national emergency call centre infrastructure. The emphasis is mainly in the enterprise and information views. An emergency service was chosen as the application, because of it's necessity in modern day society and its role in minimizing injury, loss of life and damage to property in emergency situation, and also as one alternative to providing nation-wide emergency coverage, a requirement demanded by legislation in many countries.

Requirements and ways in which of emergency services are currently offered were investigated, as well as new international developments in the implementation of such services. The shortcomings of the traditional way of offering emergency services were also discovered.

Next, open distributed systems were examined, which provide an opportunity for independent service developers to create and offer services on a telecommunication platform. This is a departure from the way services were provided before, which would only be from inside the telco infrastructure by a select group of trained technicians. This method was not only very time consuming, but also costly, and would be done only for services that appealed to a wide segment of the population and were guaranteed a long life span. Open distributed systems open up the network to any service provider, who can use the underlying functionality of the network for the

implementation of his or her application. In addition, the time taken for a service to be developed by programmers is vastly reduced. In this way, the range of services offered to users increases tremendously, creating the possibility of niche or short term services being offered, without a tremendous amount of time and resources having to be spent.

A methodology of service design, the RM-ODP, is examined, which suits the development of an emergency service in many ways. The service is a combination of information structures, in the form of databases, that are merged with a communications network. In addition, by nature, the emergency service is distributed, with the call centre, databases and emergency response personnel in various locations. The RM-ODP handles the convergence of telecommunications and information systems through its object oriented approach, and is designed for the creation of distributed systems. In addition, OSA/Parlay is used to provide the application with access to network functionality through its APIs.

The RM-ODP approaches the design of a service through a series of viewpoints, that each examine different layers and complexities of the design. The enterprise viewpoint, which is a complete high level description of the service and the role players involved, as well as the permissions, obligations and prohibitions of each role player, and the contracts between them, is fully described in **Chapter 3** UML software modelling tools such as use cases, use case diagrams and robustness diagrams are used to refine the design in the enterprise viewpoint, and to extract the objects and classes that will be used in the implementation.

The next viewpoint, the Information viewpoint, fully describes in **Chapter 4** the flows of information between the various objects in the service. As OSA/Parlay is used to enable the application to access the underlying network functionality for its implementation, the Information viewpoint mainly illustrates the flows of information between the various OSA/Parlay interfaces, that together implement the emergency service. Because of the use of OSA/Parlay, the design of the information viewpoint was constrained, due to the already defined classes and methods. Instead of creating classes and defining the flows of information between them, this viewpoint used the pre-defined classes, interfaces and methods already found in the OSA/Parlay standards.

In the Computational viewpoint, illustrated in **Chapter 5**, the classes and information flows of the previous viewpoint are decomposed into the interfaces of the various components that the application consists of, and how they communicate with each other.

Finally, the Engineering and Technology viewpoints of the design are briefly discussed in **Sections 5.4 and 5.5**.

6.2 Conclusion

The design methodology provided by the RM-ODP has proved very suitable in this work for rapid service creation in refining the design of a program to the level that coding can begin, once the standards had been deciphered. The viewpoints help to initially fully clarify the requirements of the system, and then continue refining it until it can be implemented in a programming language. The design is also implementation independent.

Further, the use of OSA/Parlay APIs in accessing the network functionality have also proved beneficial and straight-forward to implement and comprehensive in their scope. But perhaps their greatest advantage is that they allow the application to run on any platform, irrespective of the level of functionality it already provides. This is crucial for an emergency service such as this one, for it relies on advanced technology to improve its implementation. Using the OSA/Parlay APIs allows the improved functionality of this service to be offered anywhere, even in areas that have only a very basic telecommunications network.

The benefits that this design can offer over conventional emergency services are:

- allowing the caller, emergency centre dispatcher and emergency response personnel full access to all the functionality of the service, despite any limitations on their telecommunications network
- finding the location of the caller, unless the call has been made from a VoIP phone
- ease and speed of obtaining relevant information, be it the home address of the caller, or previous records of emergency calls made from the same number
- the range of backup information available to the dispatcher and emergency personnel, such as medical records and details of present medical conditions or allergies to medication
- ease and speed of sending relevant information to emergency response personnel, such as map of emergency or history of violence at a particular residence

- offering the service as a third party service provider enables a municipality or even country to offer an emergency service to its population, without having to wait for the government to provide this service

These benefits streamline the implementation of the service, making it more efficient and faster than most current emergency services, and most importantly, able to save more lives.

6.3 Further work

Based on the design of the emergency service, the following points are recommended for further work.

6.3.1 Finding location for VoIP telephones

As more and more people start using VoIP telephones, the number of calls made to emergency numbers will also increase. At present, there is no real method to find the location of callers using VoIP calls, as discussed in **Section 1.4.3**. However, in many countries, legislation is in place that demands VoIP service providers to offer the same level of 911 or 112 emergency service as is found on other telephone networks.

6.3.2 More accurate and reliable location retrieval of mobile phones

Despite OSA/Parlay having an API in their Mobility SCF that finds the location of mobile phones in an emergency, this service does not seem very reliable, nor does general location identification of mobile callers seem to be as accurate and reliable as desired. In fact, the OSA/Parlay Mobility standards [43] say that when an application wants to locate a mobile caller who has placed a call to an emergency service (see **Section 5.3.5**), the location of the caller is not always sent immediately, and an identifier will be sent to further help the application obtain the location.

Addressing this issue of difficulty in quickly and reliably locating mobile callers, the Location Interoperation Forum (LIF), recently created by Ericsson, Motorola and Nokia, has proposed to unify or align the different location technologies by devising a protocol for applications to use in requesting the location of mobile terminals

from networks. This mobile location protocol is not just for GSM, and can be used for other types of networks as well [44]. Such a protocol would be invaluable for emergency services.

References

[1] A.-J. Moerdijk and L. Klostermenn, "*Open Service Architecture: Concepts and Standards.*" http://www.ericsson.com/mobilityworld/developerszonedown/downloads/docs/parlay/OSA_concepts_standards.pdf.

[2] Official Record of the ANSA Project, "*ISO and ITU RM-ODP, Reference Model for Open Distributed Processing RM-ODP.*" http://www.ansa.co.uk/ANSATech/95/Briefing/16630001.pdf.

[3] Z. Lozinski, Senior Technical Staff Member, IBM and President, The Parlay Group, "*Parlay/OSA - a new way to create wireless services.*" http://www.parlay.org/imwp/idms/popups/pop_download.asp?contentID=6915.

[4] "*Rescue Training Resource and Guide: World Wide Emergency Numbers.*" http://www.techrescue.org/archive/emergnums.html.

[5] The Federal Communications Commission (FCC), "*Federal Communications Commission: 911 Services.*" http://www.fcc.gov/911.html.

[6] "*Telecommunications Amendment Act, 2001, Act No. 64, 2001.*" http://www.polity.org.za/govdocs/legislation/2001/act64.pdf.

[7] The Object Management Group (OMG), "*OMG Members Meet to Advance Interoperability Standards.*" http://www.omg.org/news/releases/pr2001/2001-02-12.html.

[8] http://www.nextel.com/services/worldwide/coverage/country_list.shtml.

[9] "*Helios Technology Ltd, Caller Location in Telecommunication Networks in view of enhancing 112 Emergency Services.*" http://europa.eu.int/information_society/topics/telecoms/regulatory/studies/documents/helios_final_report.pdf.

[10] National Emergency Number Association. http://www.nena9-1-1.org/9-1-1TechStandards/TechInfoDocs/FMSG-TID%20Feb2002.pdf.

[11] National Emergency Number Association. http://www.nena9-1-1.org/technical/Standards-PDF/nenamg.pdf.

[12] Micromations Systems, "*Micromations Systems, the news page.*" http://www.micromation.co.za/News.html.

[13] "*Wireless Emergency Services.*" http://www.mobilein.com/wireless_emergency_services.htm.

[14] Wikipedia, "*Voice over IP.*" http://en.wikipedia.org/wiki/VoIP.

[15] W. Emmerich, "*Distributed System Principles.*" http://www.cs.ucl.ac.uk/staff/W.Emmerich/lectures/DS98-99/dsee3.pdf.

[16] J. C. Cruz and S. Ducasse, "*Coordinating Open Distributed Systems.*" http://www.iam.unibe.ch/~scg/Archive/Papers/Cruz99FTDCS.pdf.

[17] ETSI ES 203 915-6, "*Open Service Access (OSA); Application Programming Interface (API); Part 6: Mobility SCF (Parlay 5).*" ETSI Standard V1.1.1, April 2005.

[18] J. Crowcroft, "*Open Distributed Systems.*" http://www.cs.ucl.ac.uk/staff/J.Crowcroft/ods/node3.html.

[19] Wikipedia, "*Object-oriented Programming.*" http://en.wikipedia.org/wiki/Object-oriented_programming.

[20] The Object Management Group, "*Unified Modelling Language.*" http://www.uml.org/.

[21] Kerry Raymond, Distributed Systems Technology, "*Reference Model of Open Distributed Processing (RM-ODP): Introduction.*" http://www.inf.utfsm.cl/~rmonge/seminario/RM-ODP/icodp95.pdf.

[22] F. Wai, "*Towards an Open Architecture for Real-Time.*" http://www.ansa.co.uk/ANSATech/94/Primary/110601.pdf.

[23] Distributed Multimedia Research Group, Computing Department, Lancaster University, "*Reflections in Open Distributed Systems.*" http://www.comp.lancs.ac.uk/computing/research/mpg/reflection/desc.html.

[24] ETSI ES 202 915-1, "*Open Service Access (OSA); Application Programming Interface (API); Part 1: Overview (Parlay 4).*" ETSI Standard V1.2.1, August 2003.

[25] R. Christian, "*Providing User Context Support for Next Generation Network Services*," November 2004.

[26] Z. Lozinski, "*The Parlay APIs*." `www.informatik.hu-berlin.de/~xing/Lib/ParlayAPIs.pdf`.

[27] ETSI ES 203 915-3, "*Open Service Access (OSA); Application Programming Interface (API); Part 3: Framework (Parlay 5)*." ETSI Standard V1.1.1, April 2005.

[28] ISO/IEC JTC1/SC21/WG7, "*ITU-T X.901 — ISO/IEC 10746-1 ODP Reference Model Part 1. Overview*." Draft International Standard (DIS) output from the Editing meeting in Helsinki, (Finland), 15-18 May 1995.

[29] Agile Modelling, "*Robustness Diagrams*." `http://www.agilemodeling.com/artifacts/robustnessDiagram.htm`.

[30] ETSI ES 203 915-3, "*Open Service Access (OSA); Application Programming Interface (API); Part 3: Framework (Parlay 5)*." p. 20, ETSI Standard V1.1.1, April 2005.

[31] ETSI ES 203 915-3, "*Open Service Access (OSA); Application Programming Interface (API); Part 3: Framework (Parlay 5)*." pp. 105 - 107, ETSI Standard V1.1.1, April 2005.

[32] ETSI ES 203 915-3, "*Open Service Access (OSA); Application Programming Interface (API); Part 3: Framework (Parlay 5)*." pp. 59, 130. ETSI Standard V1.1.1, April 2005.

[33] ETSI ES 203 915-4-3, "*Open Service Access (OSA); Application Programming Interface (API); Part 4: Call Control; Sub-part 3: Multi-Party Call Control SCF (Parlay 5)*." p. 13, ETSI Standard V1.1.1, April 2005.

[34] ETSI ES 203 915-6, "*Open Service Access (OSA); Application Programming Interface (API); Part 6: Mobility SCF (Parlay 5)*." pp. 16-17, ETSI Standard V1.1.1, April 2005.

[35] ETSI ES 203 915-4-3, "*Open Service Access (OSA); Application Programming Interface (API); Part 4: Call Control; Sub-part 3: Multi-Party Call Control SCF (Parlay 5)*." pp. 14-17, ETSI Standard V1.1.1, April 2005.

[36] ETSI ES 203 915-8, "*Open Service Access (OSA); Application Programming Interface (API); Part 8: Data Session Control SCF (Parlay 5)*." p. 11, ETSI Standard V1.1.1, April 2005.

[37] ETSI ES 203 915-12, "*Open Service Access (OSA); Application Programming Interface (API); Part 12: Charging SCF (Parlay 5).*" pp. 11-12, ETSI Standard V1.1.1, April 2005.

[38] ETSI ES 203 915-4-3, "*Open Service Access (OSA); Application Programming Interface (API); Part 4: Call Control; Sub-part 3: Multi-Party Call Control SCF (Parlay 5).*" ETSI Standard V1.1.1, April 2005.

[39] ETSI ES 203 915-8, "*Open Service Access (OSA); Application Programming Interface (API); Part 8: Data Session Control SCF (Parlay 5).*" ETSI Standard V1.1.1, April 2005.

[40] ETSI ES 203 915-12, "*Open Service Access (OSA); Application Programming Interface (API); Part 12: Charging SCF (Parlay 5).*" ETSI Standard V1.1.1, April 2005.

[41] The Object Management Group, "*Common Object Request Broker Architecture: Core Specification, Formal Version 3.0.3.*" March 2004.

[42] "The Object Management Group." `http://www.omg.org/`.

[43] ETSI ES 203 915-6, "*Open Service Access (OSA); Application Programming Interface (API); Part 6: Mobility SCF (Parlay 5).*" p. 41, ETSI Standard V1.1.1, April 2005.

[44] J. Zuidweg, "*Next Generation Intelligent Networks.*" p. 162. Artech House Telecommunications Library, Massachusetts, 2002.