An Integrated Modelling Framework for the Design and Construction of Distributed Messaging Systems

By Bippin Lall Makoond

A thesis submitted in partial fulfillment of the requirements of Kingston University for the degree of Doctor of Philosophy

Faculty of Computing, Information Systems & Mathematics

March 2008

ABSTRACT

Having evolved to gain the capabilities of a computer and the inherent characteristic of mobility, mobile phones have transcended into the realm of the Internet, forcing mobile telecommunication to experience the phenomenon of IP Convergence. Within the wide spectrum of mobile services, the messaging business has shown the most promising candidate to exploiting the Internet due to its adaptability and growing popularity. However, mobile operators have to change the way they traditionally handle the message logistics, transforming their technologies while adhering to aspects of quality of service. To keep up with the growth in messaging, in the UK alone reaching to 52 billion in 2007, and with the increased complexity of the messages, there is an urgent need to move away from traditional monolithic architectures and to adopt distributed and autonomous systems.

The aim of this thesis is to propose and validate the implementation of a new distributed messaging infrastructure that will sustain the dynamics of the mobile market by providing innovative technological resolutions to the common problem of quality modelling, communication, evolution and resource management, within mobile Telecoms. To design such systems, requires techniques, not only found in classical software engineering, but also in the scientific methods, statistics and economics, thus the emergence of an apparent problem of combining these tools in a logical and meaningful manner.

To address this problem, we propose a new blended modelling approach which is at the heart of the research process model. We formulate a Class of problems that categorises problem attributes into an information system and assess each requirement against a quality model. To ensure that quality is imprinted in the design of the distributed messaging system, we formulate dynamic models and simulation methods to measure the QoS capabilities of the system, particular in terms of communication and distributed resource management. The outcomes of extensive simulation enabled the design of predictive models to build a system for capacity.

A major contribution of this work relates to the problem of integrating the aspect of evolution within the communication model. We propose a new multi-criteria decision making mechanism called the BipRyt algorithm, which essentially preserve the quality model of the system as it tends to grow in size and evolve in complexity. The decision making

process is based on the availability of computational resources, associated rules of usage and defined rules for a group of users or the system as a whole. The algorithm allows for local and global optimisation of resources during the system life cycle while managing conflicts among the rules, such as racing condition and resource starvation.

Another important contribution relates to the process of organizing and managing nodes over distributed shared memory. We design the communication model in the shape of a grid architecture, which empowers the concept of single point management of the system (without being a single point of failure), using the same discipline of managing an information system. The distributed shared memory is implemented over the concept of RDMA, where the system runs at very high performance and low latency, while preserving requirements such as high availability and horizontal scalability. A working prototype of the grid architecture is presented, which compares different network technologies against a set of quality metrics for validation purposes.

Ledikasyon la kle tu la port

Soobadra Makoond

ACKNOWLEDGEMENTS

First and foremost, I would like to express my deepest gratitude to my academic supervisor, Dr Souheil Khaddaj, who continuously gave me his undivided attention and dedication throughout my thesis. I will never forget about his precious advice and encouragement, which not only was imperative for the making of this thesis, but above all, will benefit me for the rest of my life. Souheil, without you, this thesis would not have been completed or written. One simply could not wish for a better and friendlier supervisor.

I would like to thank my mentor, Dr Radouane Oudrhiri, without his knowledge and assistance; this study would not have been successful. Rad, I thank you from the deepest part of my heart for your guidance (spiritual and technological), priceless suggestions and financial assistance.

I wish to thank Dr Chris Saltmarsh, for showing me the different ways of looking at distributed systems. His advice and suggestions were most important to setup the scene of this thesis, and the arguments behind my contributions.

I would like to thank my mum and dad, Neeroo and Dilip Makoond, for their prayers, moral support and always believing in me. My sincere appreciation and love to my uncle, Raj Makoond, for always listening and offering his indisputable support since I started school at the age of 4. I express my deepest gratitude to Lalita Makoondlall, without whom, I would not have reached here.

My sincere love and thanks to my other half, Narevada Rambarassah, being my stronger half, she always kept me focus, motivated and dedicated on my work. I apologize for the many sleepless nights she had to bear with me.

I would like to thank the team, the "*InfiniBand of Brothers*", David Ong, Rytis Sileika, Michael Richardson, Paul Marjoram, Juan Amiguet and Raj Bucktowar, who collaborated on the design and build of the messaging infrastructure. Their assistance and guidance has been vital for the success of the research programme.

Table of Contents

List of Figures

List of Tables

1 INTRODUCTION

"I would like to tell you what's in my pocket. I have many things. I have a compass, a calculator, a photo album, a digital camera, a camcorder, a diary. I've got a newspaper, a video console, a pager, a movie player, a watch, a radio, a type writer. I've got thousands of business cards, a calendar and birthday cards from my families and friends. Now, you would naturally ask me how big is my pocket? Actually I've got a normal size pocket. Most probably you have the same things in yours. We both have, we all have, these many things. It is not many things; it's one! It is a mobile."

Nokia Advertisement 2007

1.1 Overview

During his work on communication and language G. Bateson pointed out that the word "**communicate**" is one of those rare words that do not have an **antonym** (Bat71). Communication begins with ***language***, the distinctive ability, which has made possible the evolution of man through models of communication within a society (Tan02). In this context, the word language is used in its generic terms, as a system of communication that uses several mediums such as sound, symbols, words, touch and the use of primitive senses. With language any message, regardless of its complexity, can be conveyed between people over a limited distance, e.g. within a room, place of assembly or across a short open space. In Victorian times "town criers" hold an annual contest to discover which of them can shout a comprehensible message over the greatest distance; the world record is less than 100 metres. Already, at that short range, a more practical alternative is to run with the message. The Roman Empire fathomed that communication model by building road infrastructures to send messengers (running or riding) across its Empire (Gib93).

Language helps in organising the message into a comprehensive format (between two or more peers) but it does not make the transport of the message faster or reach a longer distance, which is rather attributable to the medium upon which the message is transported. In fact language provides a system, rules to structure the message during a communication, whereas the medium influences the factors of reliability, performance and security, which are critical quality attributes to message delivery.

The history of communication is mankind's search for ways to improve upon shouting. This depends on optimizing the amount of energy needed and improving the security and reliability required to convey messages. For instance, when running with a message, to convey it in spoken form, it is safer to do it oneself, as sending anyone else is unreliable as the game of Chinese whispers demonstrates. The next stage of the evolution of human communication is based on a system of writing which has added a new dimension by bringing persistence which relates to communication in time, whereas the spoken form is communication in space.

In this day and age, with the development of advanced technologies and information systems; Internet and Telecommunications, the need to enhance the model of communication and comprehend all its complexities remains a challenging task. Having left the information age, to reach out for the knowledge and intelligent age, mankind has evolved to develop new communication tools. We are now in the age of wireless Telecommunication and the Internet which are **fundamentally connecting mechanisms**.

Telecommunication is compound of Greek word "*tele*" meaning far off and "*communication*" meaning transmission of information. The Merriam Webster dictionary defines, Telecommunication as communication at a distance including the technology that deals with telecommunication (Mer05). Telecommunication, hereafter Telecoms, is the transmission of signals over a distance for the purpose of communication. In modern times, the communication involves the transmission of electromagnetic waves by electronic transmitters yet in earlier centuries it may have involved the use of smoke signals, drums or semaphore (Tom05) (Wat90).

Today, devices that assist the telecommunication process, such as radio, telephone, **mobile phones** and computers, are widespread in many parts of the world. As these devices couple with the Internet, we are experiencing a new era of communication, the IP Convergence. We observe the emergence of powerful mobile devices, doubling as portable PCs, with its enhanced characteristics, changing the culture of how we communicate. In fact, Telecoms are merging rapidly with the Internet, consequently changing their business landscape, trading models, protocols and regulations (Mel05). The growing demand of the new Telecoms economy is dictating that the internet should now be in the palm of the hand (Cho02).

The Telecoms consumers are demanding more services which involve an increased mobile accessibility to the Internet, real time video transmission, real time games, Voice over IP (VOIP) and business critical transactions such as billing transactions and banking services. A report on the impact of the consumer requirements over the public services in the UK, confirms that the increasing power of the Voice of the Customer (VOC) causes major transformations of the public services (Her06) and it uncovers significant potential to harness the VOC to improve the delivery and measurement of consumer service excellence. Suddenly, operators find themselves under pressure from the market to conform to emerging specifications. To explain the evolution of the VOC, and show how they shifted the requirements of Telecoms economy the Kano model is used (Jaco99).

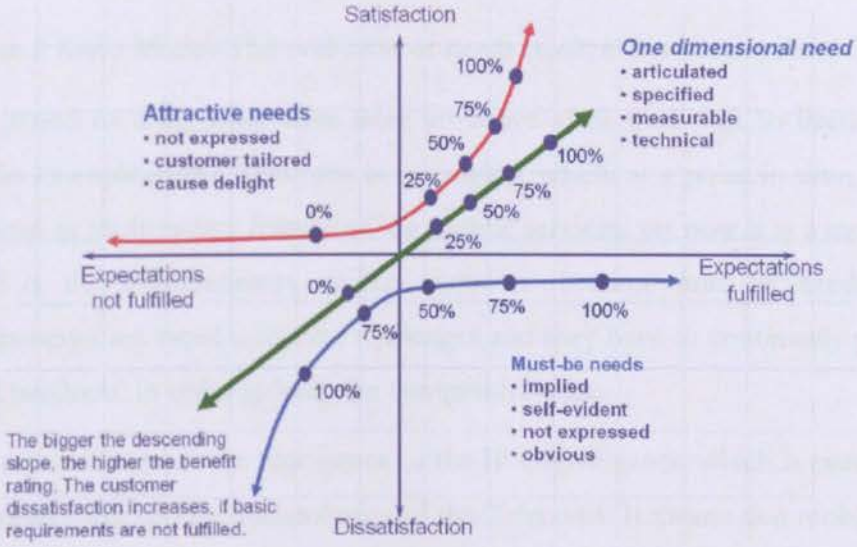

Figure 1 Kano Model (source: Six Sigma for Software, Systonomy press 2006)

The Kano model is a tool that helps to understand the VOC and divides product attributes into three categories: attractive needs, one dimensional needs and must be needs (*see Figure 1*). A competitive product ideally should meet all the "must be needs", maximises the "one dimensional needs", and includes as many attractive "excitement" attributes as possible at a cost that the market can bear. However a Kano model also shows a trend in the evolution of the needs that define the VOC, as depicted in Figure 2.

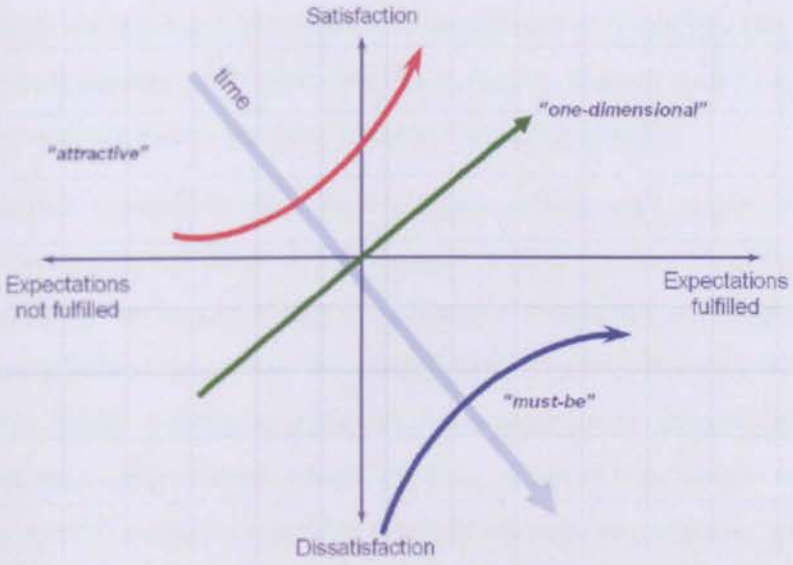

Figure 2 Kano Model: The evolution of needs (source: Systonomy Press 2007)

So, over a period of time, what were once attractive attributes, shift to become must-be attributes. An example in the Telecoms is messaging, where at a point in time, this service was introduced as an attractive feature of the mobile services, yet now it is a must be. What is observed is the differentiators quickly move to become must-be needs therefore, Telecoms operators are faced with new challenges and they have to continually provide new services and products, in order to keep the competitive edge.

Amongst these challenges is the emergence of the IP Convergence, which is causing changes to the proprietary and specific technologies of the Telecoms. It means that mobile operators are under pressure to change the way they design their telephony and messaging systems. As the mobile market moves to become more service centric, rather than technology centric, Quality of Service (QoS) has grown in essence, since in the Telecoms, innovative services are very often short lived, where the quality aspects of a system and the provided services contribute as differentiators. Decision makers of Telecoms are urging software architects, developers and engineers to build new communication infrastructure to reach for quality attributes such as horizontal scalability, high availability, economical resource management, performance and reliability. These are quality attributes that are now critical and characterise the survivability of an organisation. In many quality engineering fields, these attributes are referred to Critical to Quality (CTQs) (Bre03). The change in the Telecoms modus operandi is happening as the market waves more and more opportunities due to the expansion of the

mobile community and in an attempt to grab those opportunities, software engineers realised that traditional architectures; with their centralised chassis, cannot evolve, thus they cannot support the load and exercise in the new Telecoms economy (Cho02).

To design software systems in the very dynamic market, with continuous changes in requirements, customer perspectives and fierce competition, requires the audacity to design a system that can adapt to its environment and evolve. Evolution of computerised system plays an important role in this study. We considered Lehman's laws of software evolution (Leh80a, Leh80b, Leh85, Leh89) to construct our arguments for integrating the aspects of evolution within messaging systems, which has been elaborated in chapter 5 of this thesis. We understand that the aspect of evolution naturally led us to the concept of distributed and autonomous systems, since they lend themselves easily to adaptive and *evolutive* systems. Being loosely coupled with autonomic capabilities, distributed systems tend to scale naturally, to accommodate new features and adapt to new communication models and their environment. To reproduce these capabilities on a monolithic architecture may prove gruelling and economically not viable. Building a system that evolves brings us to the aspect of cohabitations of the "*old*" with the "*new*" (Oud02). With the emergence of IP Convergence, Telecoms are facing the problem of cohabitation; to adapt their proprietary and specific components to the IP networks and interfaces such as web services, in which distributed and autonomous systems address the problem.

When designing distributed systems, several challenges emerge. First of all, there are added complexities in managing the resources, concurrent events and the dynamic communications across distributed components. As a study on grid services applied to distributed systems explains, the continuing decentralisation and distribution of software hardware, and human resources, make it **essential** to achieve the desired quality of service (QoS) on resources assembled dynamically from enterprise, service provider, and customer systems (Fos01, Fos02). The work proposes a model for abstractions and organisational concepts that let applications access and share resources across distributed, wide area networks, while providing common security semantics, distributed resource management, performance, coordinated fail-over, problem determination services, and QoS metrics that are of importance in any given context.

Secondly, as distributed systems scale naturally, there are more and more "*conversations*", connecting interfaces and styles of communication amongst distributed participants and the ability to abstract and understand the dynamism is essential (Bat71, Chan03).

Thirdly, in messaging, each message serviced by a system is of vital importance; they have a value, cost and associated complexity. The allocation of resources to operate these messages depends on several factors and the resource management defines the economics system. Distributed and autonomous systems have complex economic models which imply an intricate task to manage and control the diverse participants of the system, in terms of resource allocation, communication management, state control etc.

Hence the significance of distributed systems has changed the problem domain of building messaging systems and by virtue of illustrating the change, we introduce the concept of **Class of Problems** which is fundamentally a categorisation process, where a class can be regarded as an information system encapsulating the problem attributes that share common properties. The reason why the classification process is based on shared features is because common problem attributes may require the same knowledge and techniques for problem resolutions. Typically a class will not only state the problem attributes, their characteristics and relationships to each other but also specify the knowledge required, and techniques available to tackle these problems. The Class of Problems is, subsequently, a multi-disciplinary process in the way it defines a spectrum of tools and techniques required to resolve a set of problem attributes.

This led us to understand that using purely classical software engineering tools were not sufficient to apprehend the problem of distributed messaging system in its entirety. Through our investigations, we found out that the aspect of communication and the interconnectivity of distributed nodes are too dynamic and too context dependent to be modelled entirely using structural design methods. However, structural design tools do add value to the design of the systems. Structural and formal methods can be used to construct distributed programs from hierarchical set of structured specifications of component instances and their interconnections (Mag95). Composite component types are constructed from primitive computational components and these in turn can be configured into more complex composite types. From an infrastructural point of view, the formal approach provides a robust and specific formalism to build composite components that facilitates the process of

managing systems structures. But, according to Garlan and Shaw, in their analysis on advances of software engineering (Gar93) (Shaw01), there is a lack of scientific rigour within software engineering, wherein structural design alone cannot exhaustively define a software problem. As a result, within this research we looked at ways of integrating dynamic modelling techniques based on scientific simulation and probabilistic methods which guided the research to the formulation of a Blended Modelling approach. This approach is based on the principles and methods of Design for Six Sigma (DFSS) (Slee06) (Crev03) (Yang03) (*see chapter 3*).

1.2 Changes in the Messaging Business Landscape

Change is inevitable except from vending machines

Robert C. Gallagher

As a study on the management of software evolution in the Telecoms argues, the telecommunications sector is being governed by the expeditious growth of two networking technologies: the wireless data transmission and the Internet (Kou01). This growth has fuelled, on one hand, the creation of new information services and resulted in a major shift from hardware to software as far as technology is concerned and on the other hand, the market is urging for new sophisticated services. The change in the Telecommunication business is occurring at management, cultural and technological levels. In the past years, during the incubation age of the mobility business, the question being asked by operators was "*we have a new technology, what service will use it?*" However, the research carried out on the mobile content revolution (SDG07) shows that in recent years, technology in the mobile market became a common place, i.e. technology is not a differentiator anymore and the question operators is now asking is "*What are the services, the Quality of Service & Service Level Agreement (SLA) required and how do I provide them?*". The mobile revolution has shifted from mobile technology to mobile content (service).

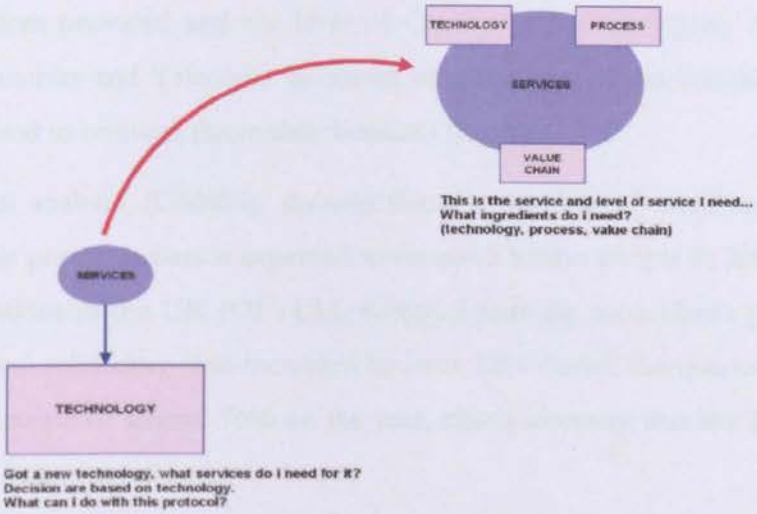

Figure 3 The shift from Technology to Service

Figure 3 explains the business shift from technological based model to a service based model, a transition that triggered the new Telecoms economy in which the mobile community operates on a fiercely competitive market. The reason behind the increase in competition is due to the increase in demand for mobile devices as Figure 4 predicts the global mobile content revenue income to the year 2011.

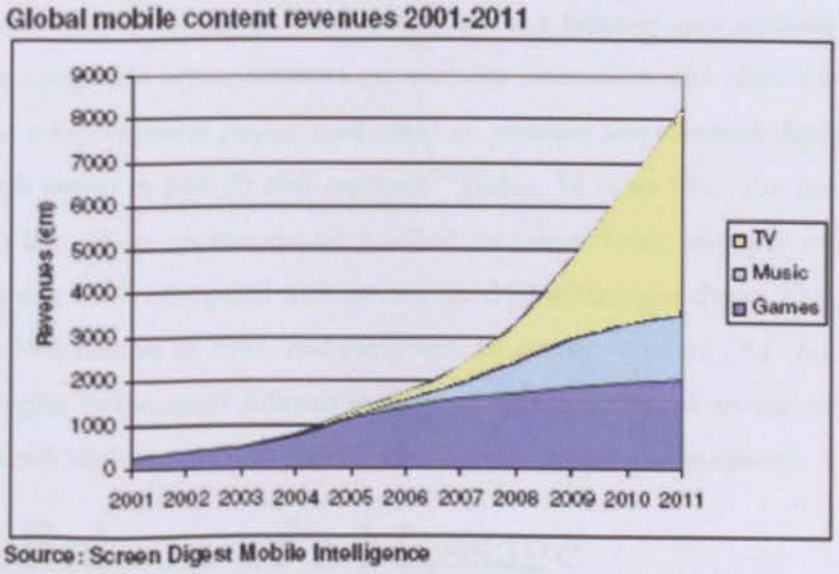

Figure 4 Global Mobile Content Revenues

Telecommunication operators, where once they were the sole owner and driver of the mobile market, see a new market emerging which is fragmented and holds several stakeholders. As a result, the operator's value chain is not found in the technology anymore

but in the services provided and the level of Quality of Service (QoS). As several major mobile manufacturers and Telecoms operators confirm in a report compiled by Intel that there is a dire need to reinvent the mobile business (Int06).

Recent statistical analysis, (GSM05), showed that the number of worldwide installed base station of mobile phone owners is expected to exceed 3 billion people in 2006. The office of Telecommunications in the UK (OFTEL), surveyed that the subscriber's growth remained strong with a total subscriber base increased by over 12% during the quarter of 2006, which represents an increase of around 76% on the year, clearly showing that the Telecoms market is growing very fast.

However, to achieve the competitive edge, Telecoms operators are looking for new application and service and a commonly used jargon is "*the killer app*", which since 1992, the best so far has been messaging, as quoted *"On Dec. 3, 1992, Neil Papworth of Sema plc (now a unit of Schlumberger Limited) used his PC to send what's generally believed to have been the first SMS message to the cell phone of Richard Jarvis, then a director at UK telecom giant Vodafone. The message, "Merry Christmas," served as a test of one specification of the emerging GSM Phase 1 standard, an initiative designed to revamp worldwide analogue cellular phone infrastructure with digital technology. Known as TeleNotes, the productized version of SMS was thought of as a technology most applicable to businesses -- for instance, as a pager-like service. Because users could only communicate with others who used the same carrier, such an enterprise-focused product made sense, as companies were commonly buying mobile phone plans from single carriers in bulk for their employees."* Today, 14 years later, the number of SMS exchanged in the UK is an average of 3 billion messages being sent per month and 100 million per day in 2006, compared with an estimated 82 million per day in 2005, 68.5 million in 2004, and 56.2 million in 2003, and each with an average cost of £0.1 (MDA06). Thus mobile messaging is the most influential element that contributed to the mobile content revolution, which was unexpected, yet changed the way people communicate.

1.3 Definition of a Message

There are numerous definitions of the word message. According the Merriam-Webster dictionary (Mer05), a message is a communication in writing, in speech, or by signals.

Message in its most general meaning is the object of communication and, depending on the context, the term may apply to both the information contents and its actual presentation.

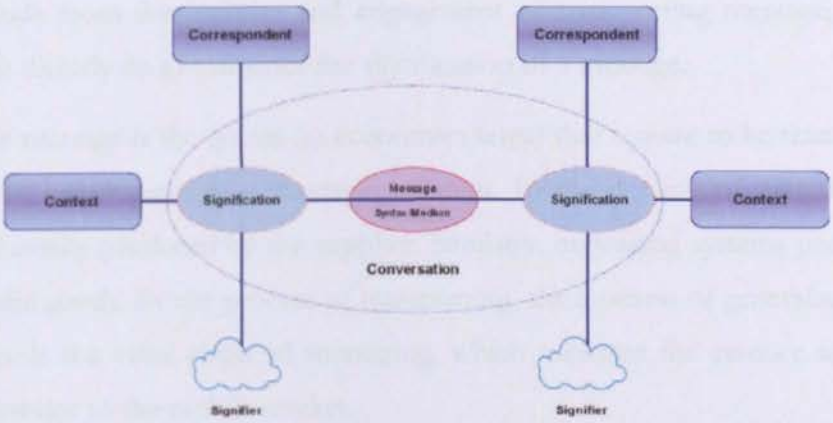

Figure 5 Conversation based Communication Model (source: (Oud02))

However on a systemic view, a message is the information sent and received that communicates various parts of a transaction, i.e. the atomic unit of information exchanged between two communicating processes. During communication a message comprises of the entire data stream including the outer envelope. According to the San Diego State university, a message is a collection of data that is ordered according to the rules of a given protocol suite, such that it is intelligible to the sending and receiving software; this is the definition, we adopt in this study.

Messages are bearer of sophisticated communication and transactional business processes and require a well defined communication model that characterises the semantics of the conversation. Figure 5 illustrates a generic communication model (Oud03) that has been adapted to illustrate a conversation based communication model. The model states that the message is not the bearer of semantics and consequently the semantics cannot be transported and can be described as follows:

Context: clear understanding of the context is critical to the business but most importantly to the economics. Messaging is within the environment of mobility, which is fundamental aspect within the context of messaging system. Despite the fact that the quality of service over wireless network is poorer that landlines, mobile phones usage boomed and gained worldwide popularity because of one quality that was not on offer to the market by landlines and that is the aspect of mobility.

Signification: signification is the process of applying meaning or sense to the communication model or more precisely to the conversation. However, in the domain of messaging, aside from the logistics and engagement of transporting message, there is not much one can directly do to influence the signification of a message.

Message: the message is the goods (in economics term) that require to be transported from one end to the other through a messaging system. In the domain of manufacturing, the goods are externally produced by the supplier. Similarly, messaging systems package, deliver and manage the goods. By the process of transporting, the business of generalised messaging systems controls the value chain of messaging, which indicates the essence and impact of this specific service to the mobile market.

Correspondent: these are the stakeholders, receivers and senders (customers and supplier respectively) to the market.

Syntax Medium: the syntax medium relates to the infrastructure, defining the medium of communication which is wireless. In the context of this study, wireless network system is an important element of the technology which signifies the mobility aspect of the service.

Rules & Syntax: rules and syntax define the structure upon which the message is built and is referred to as the system protocol. It is regarded as an information system that specifies the resources needed and the requirements to service a message during transport. Extracting vital information from the protocol will lead to the specification of the non-functional requirements or quality attributes required to handle a message. This ability allows decision makers to influence the value chain of the messaging systems.

Noise: noise is inherent to any systems, hence inherent to communication models. However, with wireless technology and mobile communication, the noise factor is increased. Unless metrics are place to control some of the noises, expected Service Level Agreement (SLA) and Quality of Service (QoS) cannot be achieved, but on the other hand awareness and control of noises can make a difference on the performance and reliability of the system providing a differentiator and edge to the market.

1.3.1 Message vs. Messaging

We believe that the words **message** and messag**ing** are sometime confused and very often used in the wrong context. By turning the word "message" into a verb, (*adding the "ing"*) the significance of the word changes and this transformation reflects the management, and technological approach of message logistics.

A message is the object that, when transported across the sender or receiver, builds up the communication. A message can have multiple formats, contents and be transported across different channels. Messaging, on the other hand refers to the **logistics** of the message. A robust messaging system has to manage the heterogeneity of a platform to efficiently communicate the various types of messages. One fundamental problem that is faced by designers of such systems is the viability of the platform to maintain the increase in message traffic and handle future message types and protocols. The ability for the logistics to adapt to changes builds up the resiliency of the system yet is a complex task to achieve.

Logistics is the science of managing and controlling the flow of goods, energy and information. The concept of logistics covers all activities relating to the procurement, transport, transhipment and storage of goods. Logistics, as generally understood, is concerned particularly with material flow (raw materials, interim and final products), but also involves providing companies with services and information through planning, execution and control. It is also the art of calculating material amounts and arranging their delivery to the proper place at the proper time to handling of the details of an operation (AFQ01).

In the messaging industry, logistics of a message covers all the business processes of planning, execution and control for handling the details of the operations dealing with 1) the procurement of content, 2) maintenance, 3) storage and 4) transportation of messages, as well as related services and information that builds up the context. As a result we understand that the business of Telecoms operators is about Messaging not message. Since the endeavour to design the "*killer app*" that will out way the TeleNotes (SMS) is an inventive task, the management of mobile message logistics is where the competitive edge of an operator can be reached.

Telecoms operators build software systems referred to as messaging gateways and messaging service systems to automate part of the process of managing the logistics of messages (IBM01). Figure 6 depicts the classical view of the process model of the message logistics which shows the Telecoms operator as the gatekeeper between the provider and the mobile user (M-User) consumers. The value chain is easily understandable and the Telecoms operators could control the value chain with their technologies and messaging gateways, i.e. the management of the message logistics.

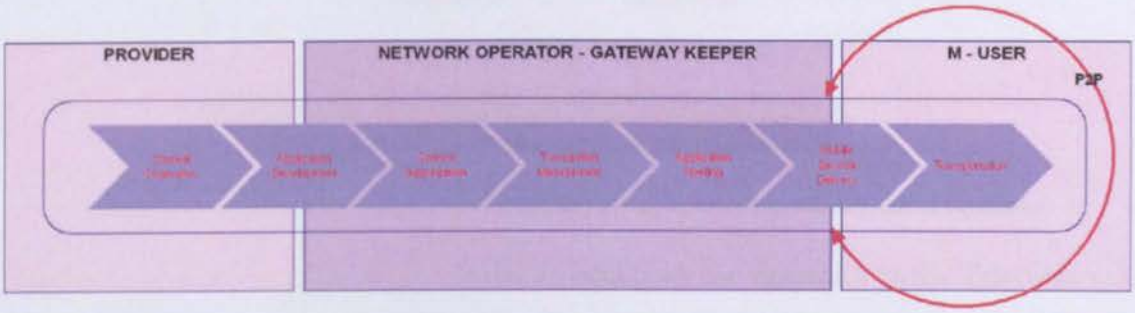

Figure 6 Classical view of Message Logistics

However, with the conquering of new technological frontiers, globalisation, the emergence of new business models and deregulations have fundamentally changed the operating parameters for the Telecoms operation business activities. In the course of just a few years, the market for telecoms services has evolved from the days of sturdy national monopolies into a dynamic competitive market which no longer allows to be confined by national borders or traditional technologies and industries.

As the market grows, increasing demand for messaging services with increasing number of mobile users, the management of the message logistic tends to become more complex. The messaging market has been fragmented with new stake holders and key players as a result the "rigid block", no longer reflect the Telecoms market model (*see Figure 7*).

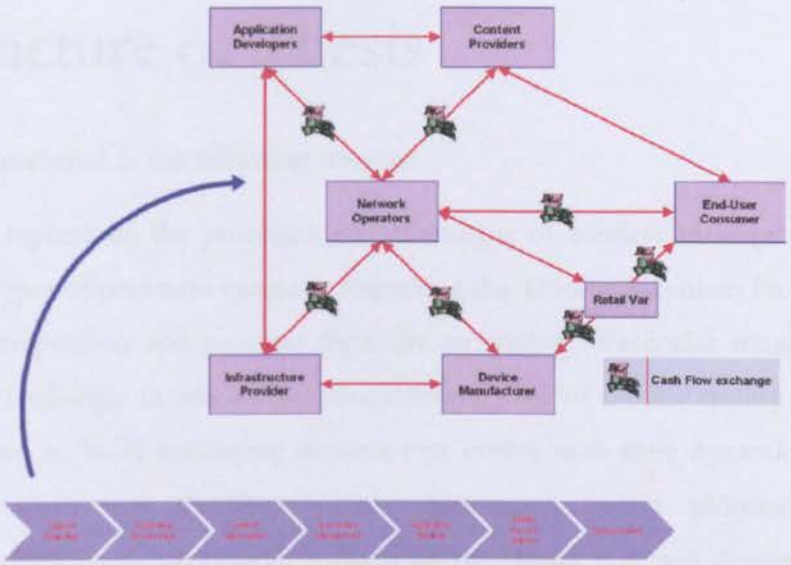

Figure 7 The shift from the value chain to a value web within the Telecoms

An analysis of the Telecoms market (Mas02), addresses the dynamics of the Telecoms value chain with the objective of identifying carriers' strategy to re-build competitive advantages in future converged networks. The single value chain is now a value web and at each "mesh", there is a cash flow exchange as shown in Figure 7. The new key players are now, content providers; developing new messaging based applications, device manufacturers and infrastructure providers. Thus, the business model of operators have expanded from a Business to Consumer model (B2C – content providers to end user consumer) to incorporate models such as Business to Business (B2B – application developer to infrastructure provider operator); primarily providing banking and financial services; Business to Government (B2G); and Business to Employee (B2E); with the upcoming services like work management, scheduling, (email and calendar). Not only has the money exchange been distributed over the value web, the criticality of each service has increased. In this new competitive arena, the versatility of the Telecoms operators is of utmost importance in this dynamic market.

1.4 Structure of Thesis

The thesis is structured in the following manner:

- Chapter 2 reports on the problems and challenges of wireless messaging which result from two types of pressures currently impacting the Telecoms market; Pressure from the business perspectives and pressure from the technology. Particular attention is paid to the quality challenges in wireless services delivery. One of the key quality attribute is the consideration to build messaging systems that evolve with their dynamic environment; hence we show how distributed and autonomous system addresses this quality challenges. Moreover, we present a **class of problems** that was devised to meet the complexities involved in building distributed messaging architectures. The class of problems is multi-disciplinary and is resolvable by practicing a blended modelling approach that combines two types of modelling techniques namely, inductive and deductive methods.

- Chapter 3 evaluates the modelling tools that form part of the blended modelling approach. At the requirement phase, we present the mechanics of the blended modelling approach with special focus on requirements classifications and styles. This process allows us to know which types of modelling tools to be used for a given style of requirements. At the design and modelling phase, we explain the concept of inductive modelling and evaluate the tools, within this category which are essential to simulate the behaviour of the systems. At the implementation phase, we discuss the network technologies upon which the messaging infrastructure is established. We provide an evaluation analysis that supports the decision made on the choice of the technologies.

- Chapter 4 shows how modelling and simulation enhances the process of software design, adding capabilities such as optimizations and predictions. We start by demonstrating how multi-agent systems of a real messaging application can be designed and dynamically simulated to derive essential knowledge which is then use as implementation directives for prototyping and development. We then, presented a case study on service discovery within distributed systems, where we modelled the different communication styles, within the messaging application. We evaluated, through the process of simulation,

different types of service discovery mechanisms and used the observations to discuss on the ones that conform to the requirements of distributed messaging systems.

- Chapter 5 presents a new algorithm called the BipRyt algorithm, which is a run time quality assurance mechanism that ensures that a given quality model of a system is preserved at all time. The BipRyt algorithm addresses the aspect of software evolution by ensuring no decline in quality as the system scales and we demonstrate how the algorithm addresses Lehman laws of software evolution. The algorithm operates by providing continuous feedback regarding the health of the system at run time. We also presented the architecture of the system and explained how it is integrated within the communication model of messaging systems. We then carried out experiments, in the context of load balancing strategies, to compare the BipRyt algorithm with the commonly used Round Robin, Response Time and Least Connections algorithms.

- Chapter 6 reports on the implementation of the knowledge gathered from the simulation exercise of chapter 4 and integration of the BipRyt algorithm of chapter 5. On an earlier study (Mak04), we designed some parts of the human conversation, focussing mostly on the aspect of conversational dynamics. The lesson learned from the behavioural characteristics of distinct processes of the human conversation was used to propose a novel communication model for messaging system. We implement the communication model and test it against predefined quality attributes through a practical case study of the Telecoms, called the Service Level Agreement (SLA) enforcement. We present the organization of a distributed messaging system as a grid structure and deployed the communication model and argue that the management of the communication model can be done through a database management system, i.e. treating the communication as a database. We built a prototype and tested two deployments of the communication model; one over InfiniBand and one over Ethernet. We demonstrated how distributed shared memory over RDMA channel interface can be exploited to build a communication model and further reinforced the model by incorporating the BipRyt algorithms as a supervisory component to address the viability of the system in terms of quality of service.

- Chapter 7 concludes the thesis.

2 WIRELESS MESSAGING: PROBLEMS & CHALLENGES

To repeat what others have said, requires education, to challenge it, requires brains

Mary Pettibone Poole

2.1 Overview

Recent investigations of the undercurrent changes of the Telecoms market suggest that the key market differentiators of the mobile industry are found in the service provided and the technological innovations, designed to facilitate the IP convergence (Dam04, Gao07, Mel05). Based on these facts we observe that the Telecoms business is being pressured by two major forces. One of these forces, initiated by the market, is urging the Telecoms operators to change their working model since there has been a shift in the way business is carried out over the last few years and this is referred to as the **pressure from outside**. The other force is attributable to the fact that the technology currently being used by the carriers also require some kind of transformation to adapt to the new Telecoms Economy and we refer to this as **pressure from the inside**.

In order to win in a very competitive market, organisations must differentiate their products from those of their competitors which is mainly caused by the outside pressure. Amongst the major Telecoms operators, there is almost no major differentiation in the technology they are using. They converge towards using the same technologies (GSM, GPRS and 3G) and the same application which are heavily oriented towards messaging (SMS, MMS and Video Messaging). To increase their differentiation, operators are campaigning to innovate and improve their service models which are leading to significant evolution in the characterization of the services provided.

The studies carried out on service innovation in the Telecoms (Gall00, Ham01, Miya05), present a sample of the major improvements introduced by the Telecommunication carriers

to their organisations. Yet these studies only cover the traits of functionalities, tools and commercial innovations and fail to discuss the quality challenges inherent to these innovations. It has been shown in (Stef93) that the fundamental differentiator that complements and reinforces new service introduction of the Telecoms is the quality of service (QoS), i.e. to what level of quality a particular service is being supplied. Moreover, the fact that there are external regulations from the increasing political influence to the Telecoms transactional model, operators are looking for more efficient and effective means to provide their services (Hor97, Int00).

As far as the Pressure from inside is concerned, IP convergence is being regarded as the next technological shift of the new Telecoms economy. A report from the Learning Initiatives on Reform of Network Economies (Mel05) states that the introduction of Voice over IP services (VoIP) has raised a number of issues of adjustment to the new environment by 1) Telecoms operators and service providers, 2) by policymakers and regulators, and 3) by the users. The report confirms that major technological improvements, which dramatically reduces unit costs and expands service capabilities, offer the potential of enormous benefits in terms of network and market expansion, cost and price reductions, and new service development. Yet it also brings the threat of significant losses to those benefiting from traditional ways of doing business, hence the obligation for the inherited structure of policies and regulations to be reassessed and modified to meet the new challenges and opportunities unfolding.

In his study of economic history, Joseph Schumpeter (Schu75) described the process of major technological change as "*creative destruction*". The convergence with IP is changing the strategic business model of Telecoms operators since they are not dealing with Telecoms specific and proprietary technological infrastructure any more. The new messaging model extends beyond the traditional SMS and MMS protocols to include the complexity of IP messaging systems. As a result, Telecoms operators are faced with the same problems as those of the IT departments, which are typically challenges of commoditisation, scaling resources, Information Technology management (IT & Cyber Parks) and efficient operational management. Telecoms operators are required to change their ways and culture of operating their business to include the aspects of economics, cost and earned value management of the IP technological models into their organisations.

Having considered the two major pressures, the Telecoms market tends to be more service centric, demanding a high level of Quality of Service (QoS) at low prices due to the consequence of fierce competition. However services are often short lived, for example, price packages and bundles are easy to copy and very soon new functionalities supplied by the operators are available to everyone and is not a differentiator any more.

Nevertheless, there are differentiators that tend to be longed-lived which are either based on having a genuine edge in customer-service standards, or by having a genuinely lower operational cost base, allowing pricing advantages to be sustained without financial damage. Over the years, Telecoms companies have tried hard to create value added differentiators. Call management services, fax, voicemail, cellular SMS, web hosting and email hosting all of which were launched as differentiators for connectivity service providers but lasted as products not as differentiators. Technology and innovative services can provide genuine, long-lasting differentiation in the Telecoms market, nonetheless pricing and QoS matter the most (Nam02).

According to a survey on Customer Satisfaction within the Telecoms market, (OFT03), QoS has a direct impact on the customer service standards and the report argues that an iterative process of QoS monitoring through consumer surveys and feedbacks will help to further improve the customer service standards. However to achieve lower operation cost base, the operator has to improve the management and the process of their systems which include their own internal "supply chain". These systems are the automated part of message logistics, such as messaging gateways and messaging service systems, which in essence is to manage the transportation of messaging across the networks.

2.2 Quality Challenges of Messaging Systems

With the worldwide installed base station of mobile phone owners expected to exceed 3 billion people in 2006 (UKWS03) and with most new mobile phones doubling as portable PCs and internet terminals, the Telecoms are urgently looking for alternative solutions to accommodate the expected surge in mobile services and mobile users. In this very dynamic

market, quality presents a major challenge to the Telecoms. We identified two aspects of quality, namely 1) QoS for traffic engineering, and 2) Quality features of the system infrastructure.

QoS & Traffic Engineering

QoS and traffic engineering is a type of quality assurance activity which is commonly practiced in software networking fields and very often referred to as "*test for quality*". QoS is strongly linked to traffic engineering, where the investigations are focussed on the methods of obtaining and measuring metrics such as 1) traffic rate, 2) congestion rate 3) transition time or latency, 4) packet loss and 5) throughput (Sriv04). These metrics are usually diagnosed during the testing phase to validate and optimise the system performance against the specifications. There are several QoS measurement frameworks to compare the relative benefits of using different types of traffic engineering mechanisms (Med02, Sun99 and Maha98). The work carried out on adaptive QoS of network traffic (Med02) considers QoS routing within the constraints of dynamic load conditions. Furthermore the assessment carried out on QoS based routing (Sun99) evaluates several QoS routing algorithms and reports on their benefits and drawbacks. Another investigation on adaptive QoS in heterogeneous network environment (Gao03) suggests that a considerable amount of QoS research has been recently undertaken but the main portion still occurred in the context of individual architecture components where much less progress has been made in addressing the issue of an overall QoS architecture for the mobile Internet. The definition of QoS has changed within the Telecoms, which is mainly due to the IP Convergence, where not only, the aspect of QoS for Telecoms infrastructures is addressed, but operators are also required to understand the aspects of QoS for IP infrastructures. For recent years, many substantial research efforts for QoS support in IP infrastructures have been made (Gue97, Cave98, Dub00 and Rose01). The study on QoS for IP networks (Gue97) addresses practical routing mechanism and QoS extensions in Open Shortest Path First (OSPF) considering bandwidth and delay constraints to provide QoS support in intra-domain. By taking advantage of the Bellman-Ford algorithm properties (Cave98) QoS routing approaches has been developed to optimise network resources and minimise hop count. These are the new network challenges of the Telecoms in their endeavour to enforce quality methods in their automated operations.

Quality Features of the System Infrastructure

Quality features for the system infrastructure falls into a different category, which we refer as "*design for quality.*" The concept goes beyond the complexity of testing the system agai specified QoS parameters. "*Design for quality*" inherently means that quality is modelled a built within the design of the system. This kind of approach is still not fully incorporat within the commercial software community and requires knowledge of more advanc quality methods which are in essence the ability to embed a quality model within the "gene of the system; such a method is called Design for Six Sigma which is discussed in chapter The main intention is to design quality into the system and to focus on some generic qualit features that we currently believe have the biggest impact towards contributing a differentiator on the Telecoms market:

Hot Deployment: is the ability to deploy new components, packed with features onto software platform at run time. As the Telecoms business develops into a more service centric model, the system has to support a flexible architecture to accommodate new services, which enable the construction of multi-featured products. This is achieved by plugging new software components onto a base architecture by devising the necessary protocol links for inter-systems communication.

Horizontal Scalability: as the Telecoms business grows, the systems that support their functions (messaging or message logistics) also need to grow to handle more consumers, and process more data, without degrading the QoS. Williams and Smith demonstrate that, as businesses expand, it is important to maintain their performance (responsiveness or throughput) (Will04). Horizontal scaling offers a solution by tying multiple independent computers together to provide more processing power which is a resilient and viable solution as far as the evolution of the system is concerned. With growing demand in the Telecoms market to cover more economies and services, horizontal scalability, which allows the expansion of computing power, is a quality attribute becoming more stringent than ever.

High Availability: in the IEEE Task Force on Cluster Computing (IEEEC06), high availability refers to the availability of resources in a computer system, in the wake of components or nodes failure. This is achieved in a variety of ways, ranging from solutions that utilize custom and redundant hardware to ensure availability, to software that provides solutions using off-the-shelf hardware components. The former class of solutions provide a

higher degree of availability, but are significantly more expensive. This has led to the popularity of off-the-shelf solutions, with almost all vendors of computer systems offering various high availability (cluster management) products. High availability is a critical quality attribute in many fields of automation, computerised transactional systems and particularly in the Telecoms. Classically, in the Telecoms arena, messaging gateways are based on monolithic architectures and designers adopt the redundant hardware approach to build the system over an active vs. passive model. Active and Passive modes mean that two deployment nodes of the solution share a common database and at any given time one and only one node is active. Should the active node fail, the passive node reads the last defined states of the failed node from the database and becomes active.

Evolution of system: in an ever dynamic mobile market, the messaging systems have to adapt to its environment and manage its workload and interactivity. It has to be flexible to intelligently allocate new resources and distribute the resources using responsive work load distribution mechanism with regards to the economics (demand and supply) of the messaging systems. To elaborate, metrics are strategically positioned to measure and characterise the demand and supply of a messaging system in terms of message throughput, message processing / service (leading to resource consumption) and delivery rate, leading to designing the system for capacity. The economics involved in this approach of design uses to the concept of cost management, utility and earned value management.

Content Agnostic: in order to handle a large spectrum of messages packaged in different protocols, a messaging system is required to be content agnostic. At the very least this is a statement that many stakeholders believe to be critical for evolution. To build content agnostic servers is not trivial as it requires design methods that are dynamic enough to adapt and change their "nature" depending on the type of incoming and outgoing message. This adds new intensity to the work load of the logistics. In classical message service system, designers use the concept of pluggable components designed specifically to react to certain protocols. However, as the infrastructure scales to accommodate more complex services and components, this approach shows drastic drop in the performance of the system. Thus, a new problem emerges that requires a consistent and adaptive communication model.

The reader would observe that most of the quality features required for the system infrastructure, "*design for quality*", naturally tallies with the mechanics of distributed systems

rather than those of monolithic systems. When we envisaged how these quality features can be implemented within the architecture of messaging systems, we found out that there are several reasons why distributed and autonomous systems are good candidates.

2.3 Distributed & Autonomous Systems

As Gruber states in his work (Grub05), "*The mobile telecommunications industry is one of the most rapidly growing sectors around the world*", which is creating a market place governed by fierce competition, and as stressed in section 2.2, quality of service and service innovations are imperative for survivability and sustainability. Amongst the many quality factors, the Telecoms require systems that have to fundamentally meet three of the most important quality attributes, which are horizontal scalability, high availability and system evolution (Will04).

Horizontal scalability is a quality attribute that has been thoroughly assessed on both centralised systems and distributed systems (Sub00). This particular study shows the flexibility of distributed architectures to scale over large heterogeneous network computing whereas a centralised system imposes a burden on resource utility in the sense that it is, sometimes, intricate and uneconomical to expand or upgrade the system. In these circumstances, when utilities are to be added, it very often requires a physical upgrade of the existing machine's capability such as memory capacity or processors' speed. Furthermore, when it comes to High availability, in recent years, we have seen several works portraying the benefit of distributed systems to sustain high availability, (Ross05, Vila04 and Ahr94). Most of these works discuss the solution of clustering to ensure high availability within a distributed architecture. Vilas et al (Vila04) presents a new technique of deploying web services with high availability features using clustering mechanism. This is based on the virtualization of the distributed web services by creating new virtual web services as requested by the clients. These clustering and high availability strategies are less costly and more effective than those of the hardware redundancies approaches, currently used for monolithic solutions of the Telecoms. The centralised solution is also reluctant to change, whereas distributed system would allow the replication of several operations and functions of messaging logistics on several dispersed nodes. These functions can be managed through

distributed cluster management software which implements service discovery mechanism and can be deployed at run time when a typical failure occurs.

As far as system evolution is concerned, carriers have to design computerised system that will adapt to new dynamics and new requirements in order to keep the competitive edge. Several works have been dedicated to the evolution of computerised system in the domain of Telecoms. The work carried out on agent interoperability (Zhan01) assessed the shortcomings of classical software design paradigm of interoperability and proposed a multi-agent design on knowledge based understanding, which enabled a sophisticated system of co-operation that can adapt to its environment unlike traditional semantic bases interoperability. In addition, the study on the management of software evolution (Mikk00) proposes a model of system abstraction to manage the evolution of Telecoms software through the concept of services. The services enable an abstract description of conceptual properties of the system disregarding their final relation to underlying software components of the implementation. As a result focus can be shifted from individual implementation components to their collaboration at varying levels of abstraction, thus providing autonomy. Another investigation carried out on the evolution of software systems within the Telecoms (Kout01) stressed on the problems of Telecoms systems to adapt to new requirements (flexibility) and proposes reconfigurable modelling primitive called coordination contract. It also presents a protocol design primitive that can support the evolution of requirements of a transactional processing system. The common feature amongst these studies is that they all inherently embrace the models of distributed systems whilst addressing the problem of system evolution. In a way distributed systems and autonomy seemed like a pre-requisite to designing a system that can adapt to its environment and evolve.

In the discussion on evolution of systems, Hugues Bersini (Bersini05) states that of the many types of systems organisations models, those that are in the form of networks and which are based on distributed architectures are the ones that survive and are economically most viable. The study on dynamic evolution (Fung04) supports Bersini's statement as the authors explain that one can easily achieve evolution in a component-based distributed system. In this paper, Hay Fung explains the abstraction of components and their connectors facilitates system structures to accommodate changes. Currently the software industry is in favour of the iterative and incremental development approach over the traditional waterfall model, in

order to achieve flexible processes that handle requirements and reduce the risk by deploying smaller changes. In such an environment, dynamic evolution provides the flexibility in implementing changes to unforeseen and fluctuating business requirements which is very true for the development of messaging systems in the Telecoms.

Evolution is a key factor to software systems, because these systems are never isolated, but implicitly constrained by their users, the communication with external systems (machines or man) and the infrastructures (software or hardware) with which they communicate (Oud02). Each adjustment or modification from any connecting peers may change or affect the internal system. The system may be provided with some resiliency and intelligence through predefined logic, to either protect itself from these changes or adapt itself to these changes, depending on the *hostility* of the environment. In other words some decision dominance may be given to the system, defining an aspect of **autonomy**, and since the working environment is non-deterministic, the system will need to adapt and re-adapt itself, changing its course many times during its life time (Oud02). Unfortunately, investments in software systems are mostly focus on designing systems based on strict protocols and precise specifications and constraints, thus restricting the systems from its external communication. If we agree that software systems are never isolated, these types of design approach (strict protocols and precise specifications) go against the natural progress of software systems. The need for evolution in the design of messaging system can be mapped to Lehman's 8 laws of software evolution (Leh84). We considered Lehman's 8 laws of evolution to guide the design and establish a framework of study to incorporate some aspects of evolution within messaging systems.

Law (1) **Continuing Change** – "*A program that is used in a real-world environment must change, or become progressively less useful*". A software system is never isolated and constantly communicates with external entities, which during the course of conversation changes the state of the software continuously. These changes cause variations within the internal system and are required to be managed. Continuous change is the basic law of evolution and is inherently implied even if it is not explicitly stated during the study.

Law (3) **Self Regulation** - Global system evolution processes needs to be self-regulating to adapt to change.

Law (4) **Conservation of Organisational Stability** - Unless feedback mechanisms are appropriately adjusted, average effective global activity rate in an evolving system tends to remain constant over product lifetime.

Law (5) **Conservation of Familiarity** – In general, the incremental growth and long term growth of systems tend to decline.

The laws 3,4,5 of software evolution, has not been the main point of focus, in this study, since they have some dependencies on other laws and are more abstract in nature. For instance laws 3 and 4, self-regulation and conservation of organizational stability respectively, are dependent on robust feedback mechanism which is law 8. Law 5, conservation of familiarity is very dependent on law 6 and 7 which are continuous growth and decline in quality. As a result, the following laws have been addressed in the research.

Law (2) **Increasing Complexity** - "*As a program evolves, it becomes more complex, and extra resources are needed to preserve and simplify its structure*". As the number of communication agents (content providers and network elements) increases, the communication model tend to become more complex and non-deterministic. Within the Telecoms, the law also applies to the message, as the content of the message becomes more complex from SMS to MMS, and video messages, extra resources are needed to preserve or simplify the structure of the systems in terms of manageability and serviceability.

Law (6) **Continuing Growth** - "*The functional capability of systems must be continually increased to maintain user satisfaction over the system lifetime*". Scalability of the functionalities is key in the design of messaging software solutions. Although Telecoms software solutions may be well structured internally using hierarchically structured class libraries, the programs have grown large and monolithic, with a high degree of interdependence of the internal modules. The size of the programs and their relatively high level, user oriented interface makes them inflexible and discomfited to use for the construction of new functions and applications. As a result, new applications usually have to be constructed at a relatively low level, as compiled programs, which are expensive and inflexible. The high degree of integration at the class library level makes it difficult to construct heterogeneous applications that use modules from different systems, since there is a lack of abstraction. This problem can be addressed by distributed systems which allow software modules to be represented as either stand alone or pluggable components in a distributed application through defined interface or resource adapters, hence scaling the capabilities of the systems.

Law (7) **Decline Quality** – "*Unless rigorously adapted to take into account for changes in the operational environment, the quality of a system will appear to be declining.*" In order to manage the quality of the system, a quality process has to be put in place to improve both the operational environment and the system itself. In many studies directed to the evolution of computerised system (Zhan01, Mikk00 and Kout01), the aspect of quality management is very rarely considered. Yet, Lehman stresses that a decline in quality influences the viability of an evolutive system and for diverse reasons this is often the case for system that evolves. In our study we incorporated the aspect of quality modelling and quality awareness within the system (at the code level). We refer to this model as **Run Time Quality Assurance** (RTQA), modelled in the shape of an algorithm called the BipRyt algorithm which is explained in chapter 5. The BipRyt algorithm was designed primarily to take into account the operational environment and steer the system, adjust its parameters to conform to the environment at run time, thus preserving the quality of the system as the system scales.

Law (8) **Feedback System** – "*Evolution processes are multi-level, multi-loop, multi-agent feedback systems.*" Feedback is a strong management tool to revise and improve a system. Since, software systems are never isolated, a robust feedback mechanism is required to validate the compliance of system functions over which the system developers have little or no control (Morr00). Due to the lack of control over the external components, the most economic development approach is to implement, obtain feedback on how the components and the interface behaves, and then evolve in the light of that feedback. In our study, we formulated and implemented an automated feedback mechanism within the decision process of resource management to create an adaptive, decision making algorithm (*see chapter 5*).

However, in order to design systems that evolve, one is also required to consider the aspect of autonomy within the distributed participants of the system. The study on the drive behind evolution of software architecture (Estu99) shows that systems are evolving under the pressure of a number of factors, which are: 1) distribution requires components to be more autonomous and to communicate through explicit means (not linked); 2) maintainability does not requires changing of the source code of components; 3) evolutivity and mobility require keeping components independent and autonomous and 4) Cost requires buying instead of building.

Autonomy is fundamentally about empowerment, giving responsibility to several parts of the system and hence these parts are governed by their own law (Bersini05). In the study on the science of the artificial life (Sim69), the author compares the natural and artificial worlds and concludes that "quasi-autonomy" from the outer environment is an essential characteristic of complex systems which are aggregations of "stable intermediate forms". Whilst the quasi-autonomous systems or subsystems maintain homeostatic relationships with their environment, they always operate within a set of environmental constraints that restrict the entity's autonomy (Oud02). Communication between the autonomous entities relate to the exchanges of information that are performed through mutual presentations. This mutual presentation of information is a very important distinction between autonomous and non-autonomous communications. Typically in autonomous systems, the presentation of information is based on a conversational communication model which is characterised by the information content, the syntactic structure, the morphologic code (the way data is coded, generally depending on the medium), the support (substratum or communication medium) and the rhythm of data exchange (the sends or receives). Yet, the nature of autonomy is constrained by organisational mechanisms and structures that are usually regarded as beneficial restrictions of an entity's autonomy in order to maintain viability and to achieve higher/social level goals. In the science of the artificial (Sim69), the author defines five 'levels' of autonomy based on the constraints to which an entity is subjected to which are 1) **No autonomy**, where an entity is told what actions to execute and it always attempts to execute them; 2) **Process autonomy**, where an entity is given a task to perform in the form of a goal state but it has some autonomy in what steps are executed in order to achieve that tas; 3). **Goal-state autonomy**, where an entity is given an external goal, which may be satisfied by a number of states, e.g. an operating system that maintains processing capacity by deciding the run-time priority of processes; 4) **Intentional autonomy**, where an entity has the freedom to decide whether or not to satisfy external goals based on its own intentions, which can be either cooperative or competitive. A cooperative entity will fulfil the request if it can and collaborate to satisfy system goal. Competitive entity only collaborates if it receives sufficient reward based on a market system with the clear objectives of maximising its own utility; 5) **Constraint autonomy**, where an entity is prepared to violate norms or even rules to achieve its goals.

In our study, two of the above levels of autonomy, **goal state autonomy** and **intentional autonomy** have been considered and implemented in the decision making algorithm for the purpose resource management within the distributed messaging system (*see chapter 5*).

The fact that our problem domain is now addressing features of autonomy, evolution, quality modelling, and distributed systems, we are required to ensure an exhaustive comprehension of the problem attributes, and organize them into distinct classes, which we refer to as Class of Problems.

2.4 The Class of Problems for Distributed Messaging Systems

As the Telecoms shift their design towards distributed architectures, they observe a change in their problem domain. Distributed systems are complex and have brought new problem attributes to the process of building messaging systems. During our investigations, we looked at these emerging attributes and created a Class of Problems in an attempt to manage the complexity of the problem domain. The Class of Problems is similar to an information system, which means that it is a system of data records, (the problem attributes), activities that process the data (the methods related to the resolution of the problem attributes) and information arranged in a given category (specific problem domain of distributed messaging systems). The class is an entity that encapsulates attributes that share common features and the reason behind the classification process is typically because common problem attributes may require the same knowledge, methods and tools to be resolved. However, a class will not only state the problem attributes and their characteristics but also specify the required knowledge, methods and techniques to tackle these problems

The Class of Problems has been established based on the challenges from both the internal and external pressures to the Telecoms (*see section 2.1*). Whilst the market and the business side revealed a service centric and quality oriented model, the technological side demands for the evolution of the system. A generic Class of Problems of distributed messaging systems (*see Figure 8*) presenting six classes of problems, (communication, substratum, dynamics,

economics and quality), which together define the problem domain of enabling, managing and operating message logistics.

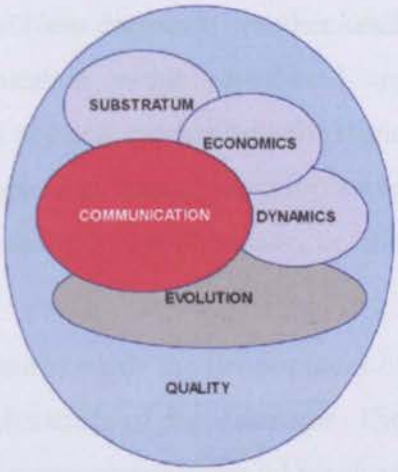

Figure 8 Class of Problems

2.4.1 Communication

Communication is the **foundation** of many systems and its understanding is fundamental to build the interaction of software entities such as objects, modules, components, agents etc. As we move from monolithic to distributed systems, the complexity of communication increases, where applications become increasingly dispersed with more connection interfaces, communication takes an ever more central role in modern software systems. Many different techniques and concepts have been proposed, in both academia and industry, for providing structure to the problems of communication in software systems (Lync02 and Dede03). Recent developed communication structures such as Remote Message Send (RMS), also known as Remote Method Invocation (RMI) or Location-Independent Invocation (LII) are used for the management of distributed object based systems (Blac90 Gros01 and Szy06). Another innovative communication structure, used for message oriented systems, is based on the message-oriented middleware that provides abstractions between the diverse communication layers (Lync02). However, it has also become clear that while such abstractions are by themselves sufficient to expose the hard problems of distributed computing, they do not implicitly solve the complexity of the communication models. The need for advanced abstraction of communication system can be compared to the evolution of database management system in which the database model and the graphical user

interfaces are examples of software elements that were once complicated to program but have, through the right abstractions and implementations, been greatly simplified (Dede03).

The question central to our problem domain is whether similar abstractions can be devised for simplifying the communication within distributed applications. Just as successful database components required not only mechanisms for storing and retrieving data, but also abstractions like query languages and transactions, successful communication abstractions requires more than the mere ability to communicate, i.e. moving beyond the models data exchange (send and receive).

Our effort in this study has been towards the development of a communication model that can adapt and evolve to the dynamics of the Telecoms. Though several works have been achieved on the abstraction of communication model for distributed systems (Alm03, Boe03 and Dede03), our work's strength comes from the approach of adopting some aspects human conversation (the conversational dynamics) as metaphor to design and build a generic communication model for the domain of distributed messaging system (Mak04). In doing so, we had to look at the elements of 1) dynamism in human conversation and 2) non-deterministic behaviour or emergent behaviour during the conversation.

The aspect of non-determinism is fundamental to the model of communication, because in employing the classical structural modelling tools such as ERD, Class Diagram and Flow Charts, to build a communication model, many factors mostly related to dynamic interaction and emergent behaviours cannot be represented. As a result we had to include dynamic modelling techniques such as simulations and prototyping in this research to depict the complexity involved when systems communicate.

The concept of dynamic interaction has been defined by G Bateson in his work on communication and language, (Bat71), to demonstrate that the behaviour of a participant can induce a specific behaviour from his peer whilst conversing. This induction process of forcing (strongly influencing) a specific behaviour from peers is present in all the communication model of all systems or organisations; yet the effect can be implicit, explicit or both. The induction of behaviour by communicating peers defines the type of communication and it presents us with another aspect of communication which is the concept of **communication styles**. We used the concept of communication styles to enrich the management of communication for the proposed distributed messaging system.

2.4.2 Substratum

According to Merriam Webster, a substratum is an underlying support or a foundation (Mer05). In our case, the class substratum is fundamentally the structure (container) utilised to carry the information being exchanged in a given communication which relates to the communication medium. The class substratum has different interpretation depending on the scale or level at which it is being observed. In computerised networking, at the lowest level, the wiring over an open network, are communication mediums, yet as we move up to the Network Interface Card (NIC) and Kernel, the medium tend to be interpreted in a more virtual manner, for instance the memory allocated by the Kernel for network copies. At the application level, or more precisely, to understand the communication amongst applications, we interpret the substratum as the type of memory in use, of which two types can be identified. On one hand we have the *"long term memory"* which relates to Storage (Databases and Database Management System). This category of medium perceives information to be stored for a long period of time often towards persistence. On the other hand we have the *"Short term memory"* which relates to the storage of information for a very short while often during a conversation; e.g. queues, stacks, array etc. This study focuses on the latter category and we understand that the communication styles influence the choice of communication medium and to illustrate, we provide the following analogy:

The little boy was asked a simple mathematical calculation – "2 X 2". Promptly the little boy answered "4". The teacher then asked for a complex integration function – The little boy answered "I need pen and paper first!"

For simple problem solving (communication style), the human mind usually allocate a small bit of memory (communication medium) to perform the calculations. But for complex problem, a different type of communication medium is required, that takes the form of long term memory (*pen and paper*). The economics behind the process of allocating resources (communication medium) for the different styles of communication play a major role in systems that need to evolve.

The concept of communication medium is of utmost importance for the design of communication models, since its abstraction is becoming more and more virtual, its management is being done at the application level. As a system becomes aware of the

distinct communication styles, it may decide on the type of substratum to use as method of storage, either long term or short term, hence an intelligent way of managing resources. This however requires a communication model which is dynamic and can also evolve.

2.4.3 Dynamics

Emergent behaviour is distinguishable when the behaviour of an entire system appears more coherent and engaged than the behaviour of individual parts of the system. Very often researchers use the analogy of an ant colony in the study of complex systems. Yet the analysis of the behaviour of individual parts (one ant) reveals very little about the systemic behaviour (Yam97). This is a problem which is also relevant to the design of computer networks. The source and effects of emergent behaviour in computer networks, focuses on several QoS parameters particularly, traffic congestion, in the attempt to understand and control the dynamic behaviour of existing networks (Yaun02). The investigation on the performance of congestion control algorithms for individual nodes over the Internet, identified important effects on system wide congestion arising from the retransmission behaviour of individual nodes (Pax99).

In a typical message system, there are many queues that communicate to each other. The control parameter of the queues, such as buffer size, no. of servers and throughput rate are usually factors that influence the dynamics of the system. In order to model and analyse these factors, one is required to adopt simulation and prototyping techniques, since classical structural model will not represent these characteristics.

2.4.4 Economics

As software architectures move towards distributed models, there is the origination of a society, wherein software entities communicate to each other and in doing so they collaborate and compete for resources. Assuming that software entities are services, consumers or resources, an economic model can be applied to the society. The economies of the society can be defined by a consumer accessing a resource or a group of resources that are made available through a service. Since resources are scarce, the problem of resource

management and resource allocation (distribution) within the society is not only a problem of informatics anymore, but also a problem of **economics**.

In recent years, we observe several studies in the field of applying economic model to distributed systems in deployments such as grid computation (Ferg96, Buyy00a, Buyy00b, Buyy01). In the study on economic models for resource management and scheduling Buyya et al (Buyy02), explored the usage of an economics-based paradigm for managing resource allocation in Grid computing environments, wherein an economic approach provided a fair basis in successfully managing distributed components and heterogeneity that is present in human economies. Competitive economic models provide algorithms/policies and tools for resource sharing within the Grid systems. There are two economic models which are based on bartering or prices models. In the bartering-based model, all participants need to own resources and trade resources by exchanges (e.g., storage space for CPU time). In the price-based model, the resources have a price, based on the demand, supply, value and the wealth in the economic system.

Classically, most of the related work in resource management for distributed systems and scheduling problems adopt the price based models (Chap99, Litz88, Berm97, Casa97 and Kapa99). In these models the scheduling components manage resources by deciding which jobs are to be executed on certain cost functions and such cost functions are often driven by system-centric parameters that enhance system ***throughput*** and ***utilisation*** rather than improving the utility of application ***processing***.

In our study we introduce a novel approach called the BipRyt algorithm that uses the principle of Run Time Quality Assurance to manage the economic model of work load distribution of distributed messaging system.

2.4.5 Quality

The current push in the Telecoms industry is focused on ensuring that software applications consistently perform as desired. Software quality is the degree to which software possesses a desired combination of quality attributes, e.g. performance, safety, robustness, reliability, security, interoperability and correctness (IEEE1061), also known as CTQs. In typical software development life cycle, most of these quality factors are assessed (test for quality) at

the implementation and testing phase. However, recently, it has been observed that much of an application's quality aspects are determined by early design decisions, in which the design choices are carried out at the requirements, analysis and modelling phases. Consequently, quality assurance have shifted into the design phase (design for quality), changing the perspective of architecting software systems. When creating a software architecture for a given application domain, system designers often justify their creation by claiming that it supports and conforms to CTQs, commonly referred to as non-functional requirements (NFR). The achievement of NFRs is attributable to many factors of the development process, such as coding styles, documentation and testing (Kazm94). However for larger and more complex systems, the achievement of NFRs, predominantly, rests in the software architecture and design, as highlighted in section 2.2.

2.5 Summary

IP convergence is affecting the ways technology is being implemented by Telecoms operators to provide messaging service. Fierce competition is driving the market, collapsing what was once a Telecoms value chain, into a value web wherein the commercials are shared amongst many stakeholders, leading to a large variety of mobile services. This has resulted in a rapid expansion of the number of mobile users and mobile data applications. Consequently, the Telecoms are required to shift the current technology to distributed dynamic and evolutive systems to cope with the expanding market. To achieve these changes, quality engineering, to both the service and the technological dimensions, is essential, which means that quality has to be modelled throughout the development life cycle, spanning from the requirement, analysis, design, to the implementation and not just at the testing phase.

To survive the expanding Telecoms market, building messaging systems with adaptive and evolutive abilities is becoming more and more crucial. An important characteristic of evolution is the problem of co habitation which, in our case relates to IP convergence, where the classical messaging systems (the old) need to co habit with new IP infrastructures. To model the aspect of evolution into messaging systems, we explored Lehman's 8 laws of software evolution which led to the conceptualisation of autonomous participants. These autonomous entities, working on a distributed environment, tend to naturally conform to the

framework of building adaptive and evolutive systems. These systems enable the abstraction of components from their connectors, thus facilitating the structure to accommodate changes. The fact that we regard distributed system as a group of distinct autonomous entities working together, introduces new complexities.

In order to understand the complexity, we devised a model that we refer to as a Class of Problems. The Class of Problems is primarily **multidisciplinary** as we observe that some classes require different approaches of modelling. For instance, within the class Dynamics, one of the requirements is to model the collaboration between software entities (components or agents) and analyse how they join and leave communication sessions. Using purely collaboration or sequence diagrams from UML is not sufficient due to their static characteristic. As a result we need to use dynamic and simulation modelling techniques, in order to investigate the model of communication from a systemic and holistic perspective.

From a quality engineering perspective, test for quality is not adequate anymore to ascertain the consistency of quality in distributed messaging systems. Within the class Quality, we are required to look at modelling tools, statistical and probabilistic techniques to measure, analyse and model quality of the system. Moreover, the class Communication demands a larger spectrum of modelling tools, ranging from structural modelling to dynamic modelling and simulation of communication amongst the peers to analyse the relationship between communication styles over the communication medium and most importantly the significance of non-determinism within the communication.

The classical modelling approach, UML, ERD, flow charts and DFDs are principally structural and static in nature, but they add value by providing the formalism required to explain the structure of a system, the design and the flow of process. However, distributed messaging systems are social networks, with non-deterministic behaviours, which only can be modelled by complementing the structural modelling techniques with the dynamic modelling approaches. Therefore, in our study we have to adopt a blended modelling approach which employs tools and methods aspired from both the structural and dynamic arenas, as the chapter 3 reports.

3 TOOLS & TECHNIQUES

"It is tempting, if the only tool you have is a hammer, treat everything as if it were a nail."

Abraham Maslow

3.1 Overview

According to Godel incompleteness theorem, a complex system can be defined as one that can only be modelled by an infinite number of modelling tools (Chai71). The development of distributed systems in domains like telecommunications, industrial control, and business process management represents one of the most complex construction tasks undertaken by software engineers (Jenn01) and the complexity is not accidental but it is an innate property of large systems (Sim96). To understand the complexity of building distributed messaging systems, we formulated a class of problems (*see chapter 2*) which shows that many types of modelling tools are required to represent the behaviour of such systems. In distributed systems we see the emergence of additional behavioural complexities since logical operations may require communicating with numerous nodes and sending hundreds of messages in parallel. Distributed behaviour is also more varied, because the placement and order of events can differ from one operation to the next.

In any complex system, there are at least two systems worth modelling; 1) the system to be developed (**the Target**), and 2) the process for developing that system (**the Process**). Both the Target and the Process require modelling and in fact within software engineering, all deliverables, from requirements, through architecture and design to the code implementation are actual models, this including the final product itself (i.e. Software or an Information System). To model the Target and the Process, we need to address the following challenges:

1. **A Multi-disciplinary Class of Problems** - as we discussed earlier, the class of problems is multidisciplinary suggesting that there are several ways of modelling the problems attributes and we were required to combine several of these approaches and models, as Figure 9 shows:

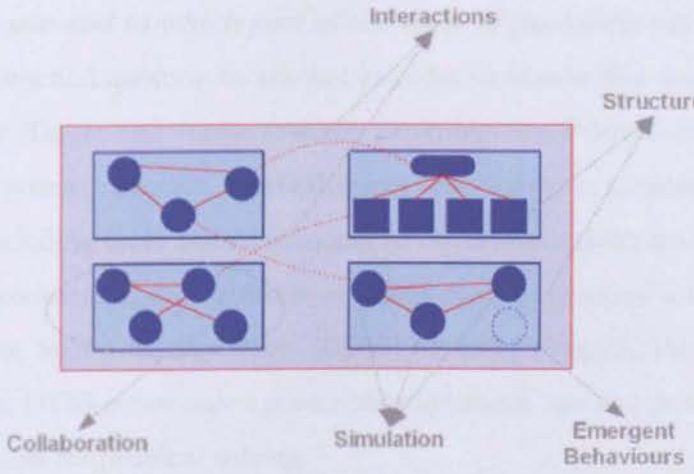

Figure 9 Multi-disciplinarism in Modelling

There are two types of modelling approaches, inductive and deductive, within the field of software engineering.

- Deductive Modelling includes the aspect of structural, functional and collaborative designs and are commonly used in software engineering, such as Class Diagrams, Sequence Diagrams, Object Diagrams, Entity Relationship Diagrams (ERD), Data Flow Diagrams (DFD), Flow Charts, Use Cases, etc...
- Inductive Modelling, are critical dynamic modelling techniques that primarily characterise the aspect of non-determinism within a system mainly arising from the occurrence of emergent behaviour. Commonly used techniques are simulations and probabilistic models of the software artefact. As we stressed earlier there is a growing concern to promote the use of quantitative and statistical methods by practicing software engineers. Statistical techniques are relatively new to the software industry whilst there is significant uncertainty in the community about their difficulty and applicability.

Considering the wide spectrum of modelling techniques, we are required to unify them in order to capture the several facets of the software system and demonstrate the power of modelling to develop software artefacts of high quality.

2. **A Blended Modelling Approach** - due to the fact that there are many modelling disciplines, the crucial question one will ask is "***how do we know which modelling***

technique to use and to which part of the class of problems are they applicable?'
This is a fundamental question to ask and in order to answer this question one needs to go "*above*" the Target and move towards exploring the Process that will be used to develop the system. Process methodologies provide the necessary guidelines and framework (including tools and techniques) to undertake a given project and to achieve high quality products. In this research we adopted some parts of a methodology called Design for Six Sigma methodology (DFSS) (Slee06, Yang03, Hay05). Unlike other methodologies, DFSS is not only a prescriptive approach but also provides statistical and probabilistic tools for problem solving.

Based on the fact that we need to evaluate the techniques of designing both the Target and Process, this chapter is organised in two parts; the first part discusses on the techniques of Software Process modelling and the second part discusses on the modelling and implementation environment of the Target.

3.2 Software Process Modelling

The development of distributed messaging system is a complex activity with a large number of quality factors involved in defining success. Despite the fact that inductive modelling is scientifically thorough for analysing and building quality engineered systems, it brings additional cost into the development life cycle. Hence, a development process should be able to blend inductive and deductive modelling techniques, to adjust the equilibrium between cost (time resource) and quality. As a result, the field of software process simulation has received substantial attention over the last twenty years. The aims have been to better understand the software development process and to mitigate the problems that continue to occur in the software industry which require a process modelling framework and in our study we employ DFSS.

3.2.1 Design for Six Sigma (DFSS)

Unlike other process modelling frameworks, DFSS is not limited to deductive and static modelling techniques, as DFSS uses inductive approaches and in some cases non-deterministic models, that are well known within the academic world, but not yet of a

common use within industry. In fact DFSS acts as a natural "*glue*" to blend the various modelling approaches providing a process framework with the primary objective of removing the characteristic of "*ad-hoc*"ness within the Process (*see Figure 10*). Using DFSS, different types of modelling tools have been integrated at the design phase of this study starting with dynamic modelling tools (inductive modelling) such as Coloured Petri Nets (CPN) and prototyping, then moving to event based modelling tools such as State Chart Diagrams and Sequence diagrams and finishing with structural modelling tools (deductive modelling) such as class diagrams.

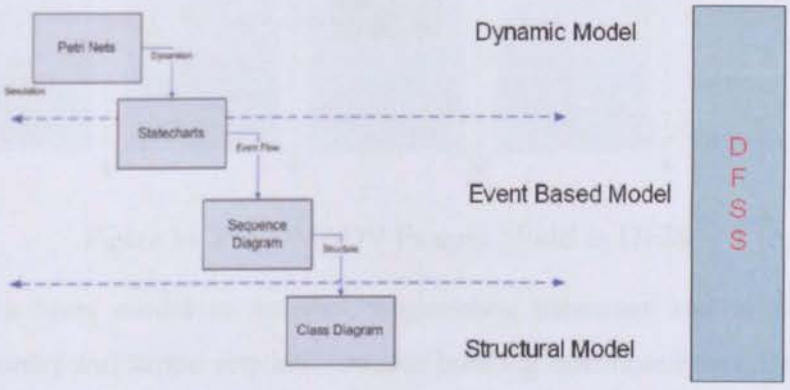

Figure 10 Modelling Translation

DFSS is also an ideal candidate to implement quality modelling procedures into the development framework. It provides traceability matrices such as House of Quality (HoQ) and Analytical Hierarchy Process (AHP) to examine the relationship between functional requirements and the desired CTQs (Non-Functional Requirement) so that quality model(s) can be embedded to the system, based on the voice of customer (VOC) and the Voice of the Market (VOM).

The DFSS process model generally follows the phases of Define, Measure, Analyse, Design and Verify (DMADV (Yang02)). The Define phase clearly states the problem and identifies the big Y's (problem attributes). We can define the Y's but we cannot acts on the Y's directly, because the Ys are dependant on other attributes. The Y's are the outputs and we need to identify and understand the inputs, which are the X's. During the Measure Phase we collect facts and gather data about the X's. There are several techniques provided by the DFSS methods to achieve such task. During the Analyse Phase we analyse the interactions and develop quantitative models of the system (process or product) or define the basis for

developing predictive capabilities through probability and simulations. The Design phase consists in developing solutions and selecting the best. Again quantitative modelling is powerful tool during this stage especially for DFSS projects. Figure 11 provides an overview of the DFSS process model.

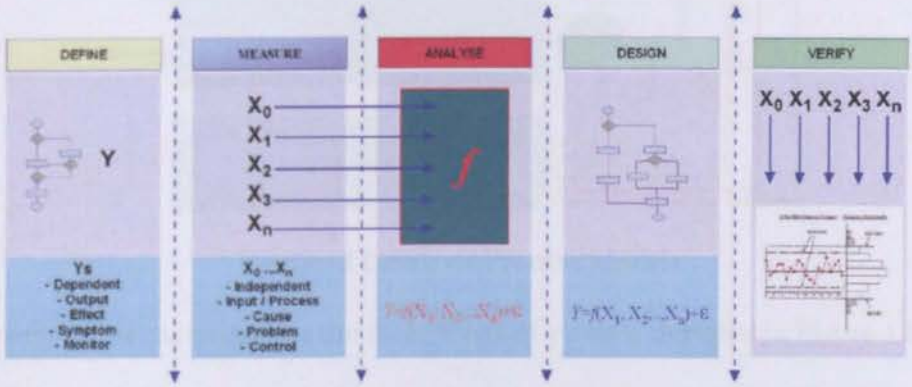

Figure 11 The DMADV Process Model in DFSS

DFSS acts as a Meta model to software engineering processes and in our context, it formalises the order and action required towards building distributed messaging systems. It provides the rules, policies and instructions that guide the research and development programme.

3.2.2 Research and Development Process

When we started our research, we were essentially focussed on the technological aspect of the problem, i.e. the technical aspect of modelling and the Telecoms technologies. However, at some point we hit a barrier since we found ourselves using several types of modelling disciplines (inductive and deductive) and it was difficult to align them within the Research. We realised that we needed a Research and Development Process model to explain how the research is to be carried out using a Blended Modelling Approach centring on our Class of Problems. We have identified three levels of processes within the context of software development as Figure 12 depicts.

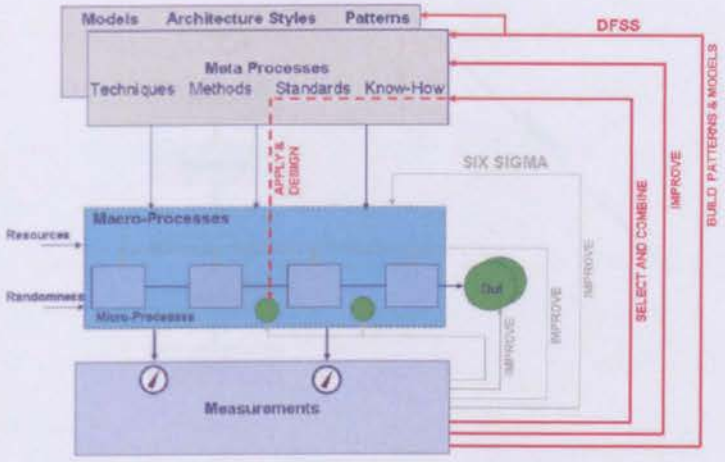

Figure 12 Levels of Process Models

The following sub sections explain the three levels of process depicted in Figure 12.

Micro-process: a project team's policies, procedures, and practices for achieving an artifact of the software process. The focus of the micro-process is on achieving an intermediate product baseline with adequate quality and adequate functionality as economically and rapidly as practically possible. In our context, these are the distinct phases such as requirement, analysis, design, simulation, prototyping, and testing.

Macro-process: a project's policies, procedures, and practices for producing a complete software product within certain cost, schedule, and quality constraints. The focus of the macro-process is on creating an adequate instance of the meta-process for a specific set of constraints. In our context this applies to the software life cycle such as water fall model, spiral model, V - shape model and incremental model. The decision over the selection of life cycle has been done at the prototyping phase and specified within the directives list given to the development phase.

Meta-process: an organization's policies, procedures, and practices for pursuing a software-intensive line of business. The focus of this process is on organizational economics, long-term strategies, and software ROI. In our context, the Meta process is the research and development process model, elaborated in the following sections.

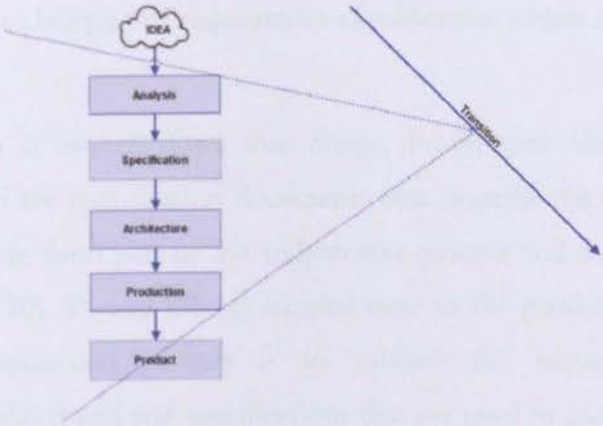

Figure 13 Research Life Cycle

The diagram in Figure 13 illustrates the Research life cycle of our study. The line, transition, shows the area of focus of the process model that spans from analysis, to specification through architecture to production. Each of the phases are drilled down (zoom in) and illustrated in the following sections.

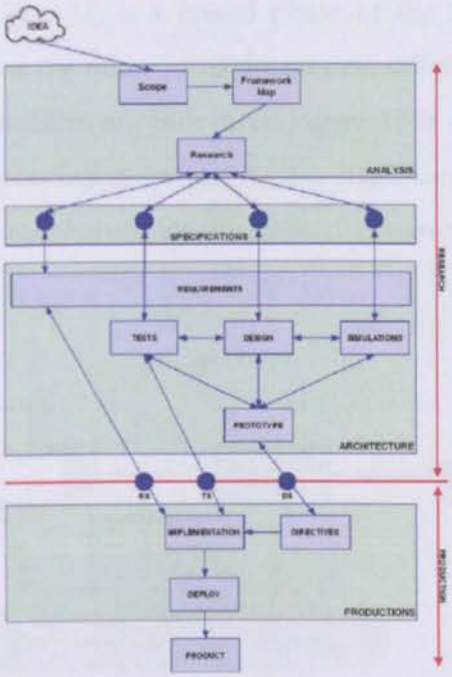

Figure 14 Research Life Cycle Expanded

Figure 14 is a magnification of Figure 13 and it reveals the clear separation between the research domain and the production domain. At the end of research phase, there are three deliverables, referred to as RS which are the requirement specification, DS relating to development specifications and TS which are the test specifications. The process of RS is

carried out using the techniques of requirement classification which is elaborated in section 3.2.3.1.

The Analysis process is broken down into Scope, Framework Map and Research. The outcomes of Research are specification documents that describe the design, simulation and test phases. These three form part of the architecture process and at the end of this phase, the three deliverables RS, TS and DS are handed over to the production process. The key objective of the production process is to validate the requirement specifications, development specifications and test specifications that are used to guide the implementation and deployment of the product. The validation process comprises of simulation and formal verification techniques and statistical modelling from the DFSS methodology. The following section explains the each distinct component of Figure 14.

Framework Map Model

The Framework map (*see Figure 15*) is a crucial phase of the Research and Development process model. This is because the outcome of the process will provide a problem domain to the research project. Each problem attribute from Figure 15 is categorised into the class of problems of the distributed messaging system. At this stage, the framework map provides a high level model of the problem domain for the research programme.

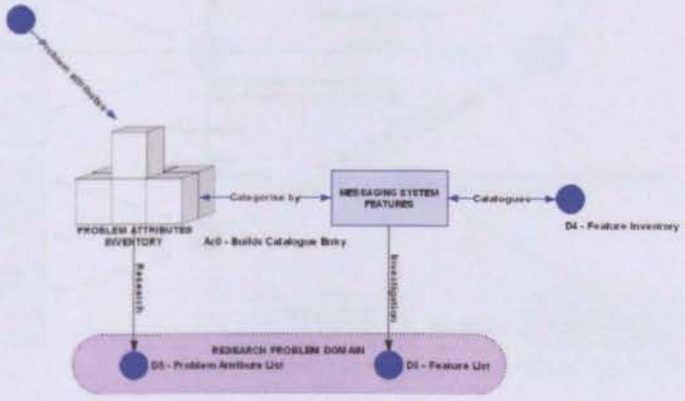

Figure 15 Research Framework Map

Research Process Model

The Research process, in Figure 16, initiates when a high level description of the problem domain is completed and this has been achieved in the Framework Map process. At this

point it is essential to clarify that the research and development process model is a learning exercise and the primary objective of the research is to analyse and question the problems, in order to understand them. The researcher looks at academia and other expert fields to understand how people are attempting to solve or solved similar problem. Evidently, this is not a linear process as one needs to abstract and build analogies and metaphors in order to reinforce his/her knowledge about the problem domain. Literature reviews are required to learn about theories and models that exist in academia. This will help in formalising the problem domain to include mathematical and formal models. This exercise helps to reduce inconsistencies and ambiguities from the problem domain. During the course of the research we build an inventory of tools, technique, adopted concepts and methodologies which is reviewed and updated on a regular basis. It builds a strong foundation of tools and methods for future research and development. Validation and reinforcement of the problem domain is achieved by mapping the problem attributes with the class of problem of distributed messaging system. In doing so, one is able to specify the distinct requirements for the research programme, which is modelled using the technique of classification of requirement styles (*see section*).

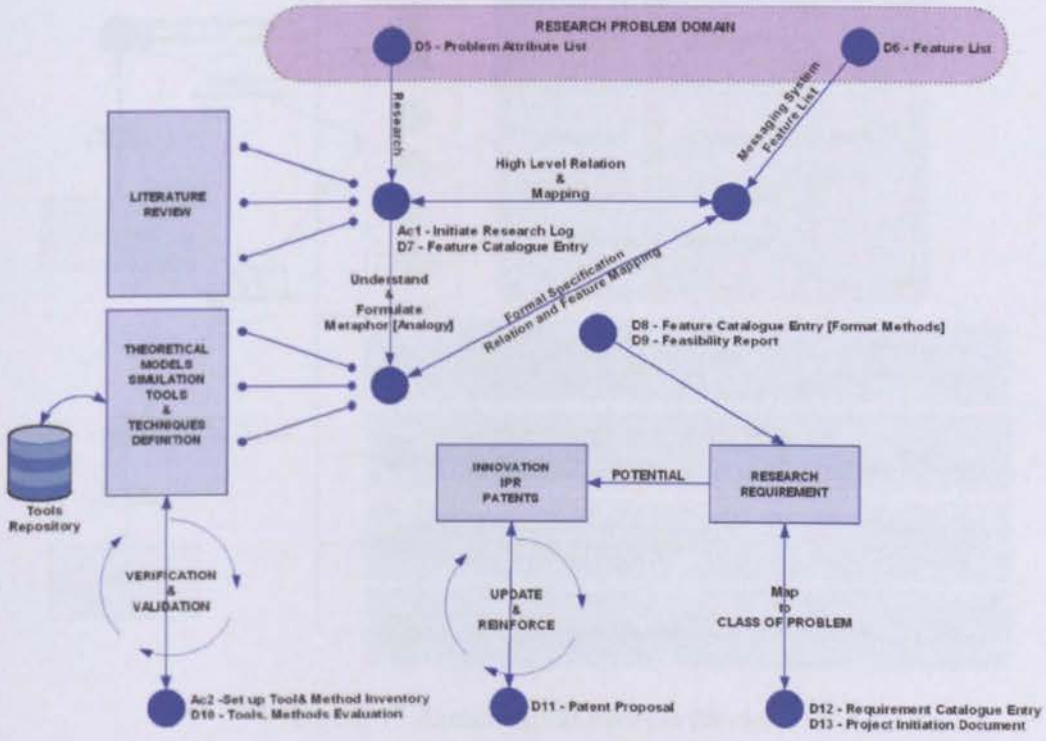

Figure 16 Research Process Model

Architectural Process Model

The Architecture model (*see Figure 17*) has for objective to validate and confirm the requirement specifications that were delivered by the research phase. In doing so, knowledge is gained to design the model specification and the test cases which are made possible by adopting methods of DFSS, to categorise the requirements into styles. Following the guidelines derived from the requirement classification technique, those functional requirements are modelled accordingly, i.e. those requirements that show properties of **data and structure style** are modelled using structural modelling techniques an those that show properties of **communication style** (dynamism) is modelled using simulation tools and run against the CTQs using the test cases.

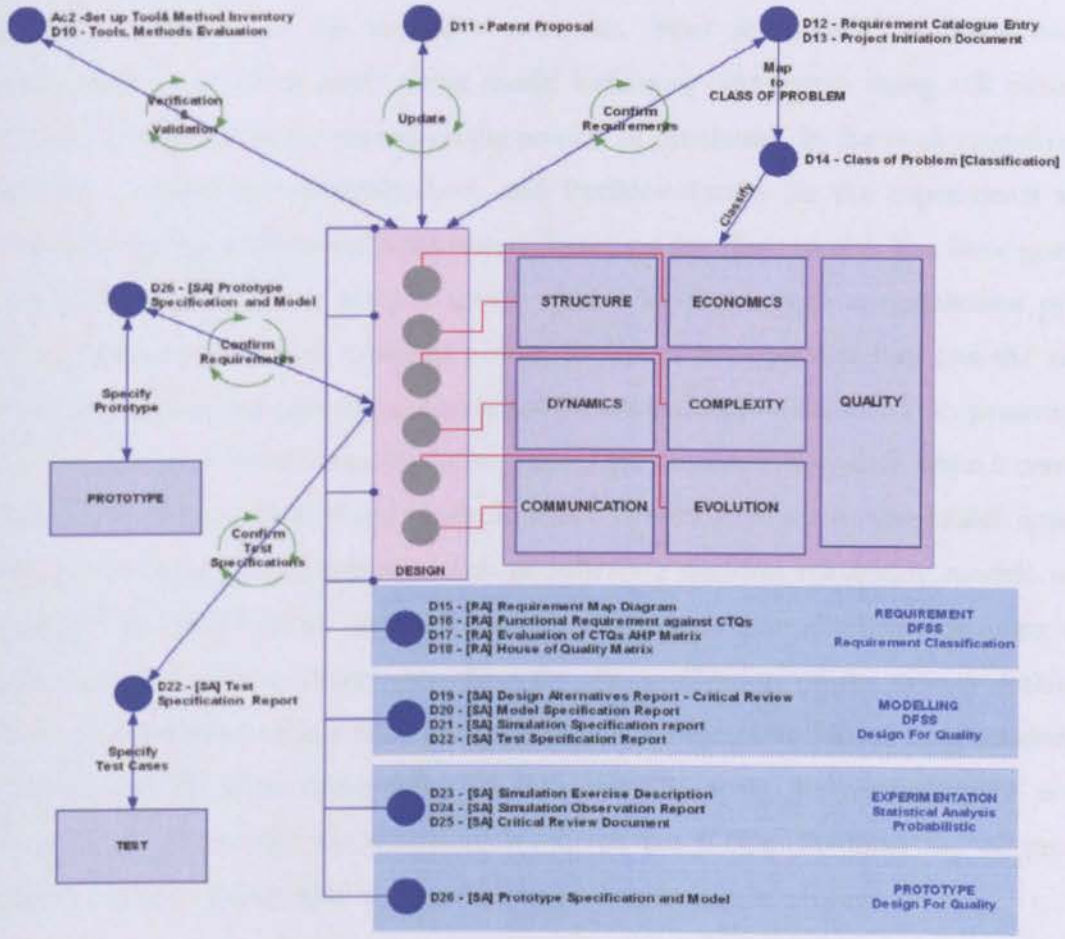

Figure 17 Architectural Process Model

The criteria and specification for the test and simulation runs are formulated by evaluating the functional requirements of the system against the CTQs using tools such as the House of Quality (HoQ) which is part of the Quality Function Deployment (QFD) family and the Analytical Hierarchy Process (AHP) (*see section 3.2.3.2*). The simulation results provide further knowledge on the system to fine tune or calibrate the models, hence validating the models and the test cases. Statistical analysis can be implemented to analyse the simulation results which are primarily used to validate the requirement specification, the design specifications and the test specifications.

Prototype Process Model

At the Prototyping phase (*see Figure 18*), one is required to build prototypes based on knowledge gained from the simulation exercises. Since simulation has limitations for decision makers to "*learn more*" about model behaviour (simulation being still virtual), a prototype is implemented to extend on the process of simulation. In the work carried out on limitation to simulation (Levy90), Levy and Pavlides discuss on the experiments which concern the profiling of a distributed system, based on the client model. The basic goals are to compare the simulation results, against quality attributes such as parallelism profile, synchronization profile and speed-up profile, produced by a profiler tool and the similar results produced by the execution of an actual prototype implementation. They present both kinds of results and demonstrate the limitations of the simulation approach when it comes to emulating the environment of real life deployment. However the reader should also note that there are advantages to simulation, such as validating decision for quality models within design and the provision of graphical modelling interfaces instead of programming code, which helps abstraction. Prototypes allow for the verification of the system within the environment variables of real platform deployment. This enables further confirmation and verification of the three deliverables, RS, DS and TS. Again the reinforcement process continues, to ensure that the specifications match the CTQs. In doing so, we produce directives for each deliverables and hands over to the production process.

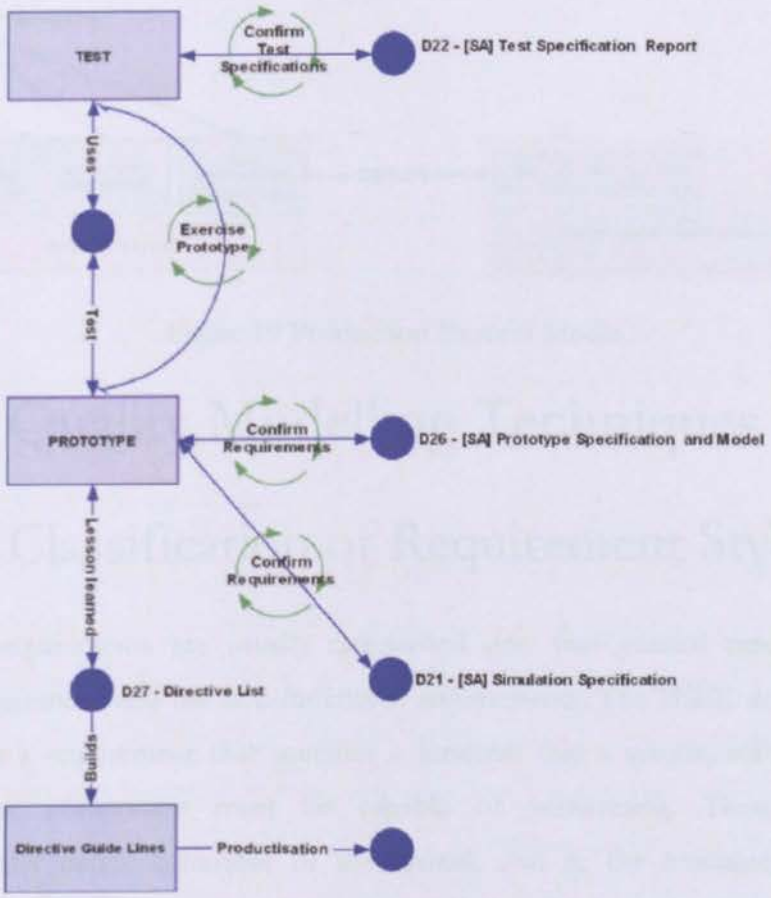

Figure 18 Prototyping Process Model

Production Process Model

This phase is outside the scope of this study where research has been completed and it is hand over process. The researcher has to produce the requirement specifications, the model specifications and the test specifications to the business analysts, the development and test teams respectively. Since quality has been addressed within the problem domain and mapped to the architectural model of the distributed messaging system at an early stage of development, this approach relates to ***design for quality***. The researcher, architects and the product manager will supervise and advise during the course of production. The directive list will also include the selection of the most appropriate life cycle(s) (usually more than one) such as the waterfall; spiral; incremental; prototyping and V model life cycles.

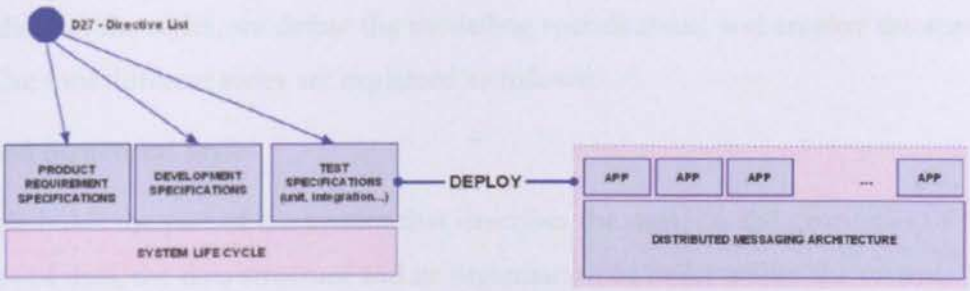

Figure 19 Production Process Model

3.2.3 Quality Modelling Techniques

3.2.3.1 Classification of Requirement Styles

Customarily, requirements are usually categorised into two general types; there is the functional requirement and the non-functional requirements. The IEEE defines functional requirement as a requirement that specifies a function that a system/software system or system/software component must be capable of performing. These are software requirements that define behaviour of the system, that is, the fundamental process or transformation that software and hardware components of the system perform on inputs to produce outputs. Non-functional requirement, on the other hand, in software engineering is a requirement that describes not what the software will do, but how well the software will do it, e.g. the CTQs of the system, software performance; software external interface requirements, software design constraints, and software quality attributes.

In our study, we extended on these two general styles to propose a new requirement framework that is composed of four different requirement styles. We have broken down the functional requirement styles into 1) the data and structural style, 2) functional and behavioural style and 3) the communication style. The non-functional requirements are referred to as the quality attribute style in this study.

Since, the class of problems has many dimensions and we need to identify each of these dimensions so that the right modelling tools and techniques are applied for the right problem attribute. We created a classification technique that categorises the problem attributes in different requirements styles; hence the selection of modelling techniques to represent the requirements can be based on the characteristics or styles of the requirements.

Depending on the styles, we define the modelling specifications and employ the appropriate tools. The four different styles are explained as follows:

Data and Structural Style

This style holds the part of the system that describes the data, i.e. the properties of the data, the types of data, the data structure and its organisation or order within the system. The task of the data styles is to primarily **separate** the **data** from the **process of treating of data**. The classical modelling tools available to handle requirements within this style are entity relationships diagram, class diagram, object diagram and component based diagram.

Functional and Behavioural Style

This style defines the functions of the system without any regards to the class these function belong to. The main objective of the functional styles is to describe the input, the process and describe the expected output of a process. The behavioural part requires tools such as state charts and CPN that illustrate the change in state of a program when a particular transition of that program has occurred. Classical tools of the functional and behavioural styles are pseudo code, B machines, Z notations, collaboration diagrams, flowcharts etc.

Communication Style

The communication style handles the complex features of system interactions than entail both the inter-communication amongst external systems and the intra-communication of internal components. The style provides the specifications of modelling the aspect of communication to capture how the different parts of the systems communicate using different styles of communication and the type of communication medium in use. These modelling tools are usually dynamic that naturally offers themselves to simulation. The dynamism is essential to describe emergent behaviours within the communication model. The tools to model the attributes of the communication style are simulation tools such as CPN, Estelle, SPIN, and prototyping.

Quality Attributes Style

Quality Attributes Style incorporates methods of modelling the non-functional requirements, i.e. the CTQs. According to Somerville, (Somm01), different kinds of non-functional requirements exist; ranging from product requirements, organisational requirements to

external requirements. The product requirements specify the behaviour of the product, e.g. execution speed, robustness and reliability. Organisational requirements are originated from the organisation's policies and procedures, e.g. process standard used and implementation requirements. All requirements that come from external factors to the system and its development process belong to the external requirements. One type of external requirement is security. However as explained in the discussions on software requirement engineering, (Somm01, Lau02, Krem98), the problem with non-functional requirements is the difficult to verify and very often they interact and conflict with functional requirements and with each other. The DFSS methodology addresses the problem of quality modelling enabling software analysts to translate functional requirements and CTQs into traceability matrices that can be use to formulate statistical models, transfer function for the analysis of the tolerance factor. These tools are quality function development; Quality Function Deployment (QFD); House of Quality (HoQ) and Analytical Hierarchy Process (AHP).

Each style corresponds to a particular view of the system and address a particular aspect of the Class of Problems. However, we observed during the study, the different views are not independent but interdependent. The multiplicity or variety of styles helps to enhance understanding the class of problem so as to reduce ambiguity. Figure 20 is a high level model of the application of DFSS to classify requirements. For a given class of problem, the definition of each style is composed and the requirements are analysed and broken down into their distinct style.

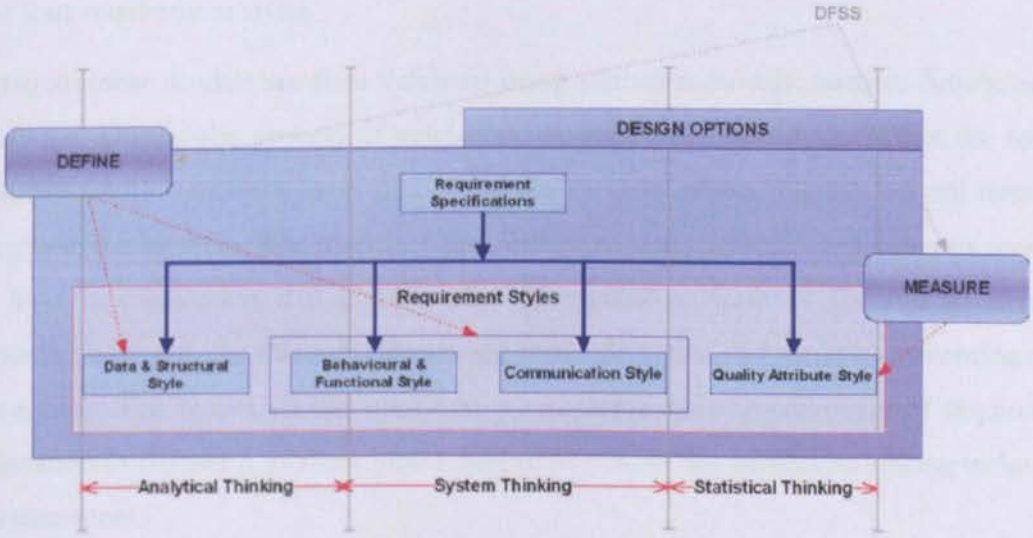

Figure 20 Requirement Styles

Figure 21 illustrates how we applied the method of requirement classification in to the problem of building distributed messaging system.

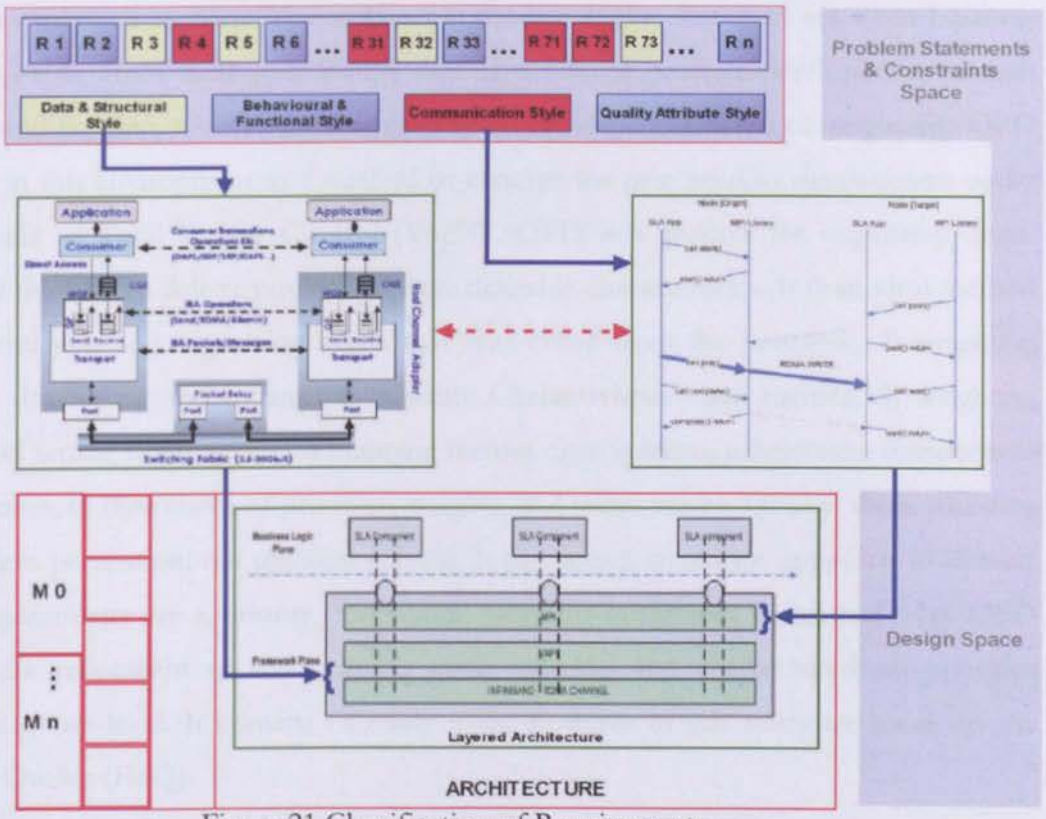

Figure 21 Classification of Requirements

Typically R_1, R_2, R_3 ... R_n are requirement statements, usually written on a requirement catalogue entry (Rose92). These requirement units are classified into models that form part of the four requirement styles.

The requirement models are then validated using statistical models, transfer functions and simulations. During the process of validation, we gain more knowledge about the models and consequently reinforce the structure of the models, which requires several iterations throughout the lifecycle. The model of the Architecture in Figure 21 is a superset model, a high level representation that conveys the information content of the related elements comprising a system, the relationships among those elements, and the rules governing those relationships. The reader should note that we implemented the technique of requirement classification in chapter 6 to show how it helped to choose the various modelling techniques per requirement.

3.2.3.2 Quality Function Deployment (QFD)

QFD was conceived by Akao Yoji in Japan in the late 1960s, during an era when Japanese industries broke from their post-World War II mode of product development through imitation and copying, towards moving to product development based on originality. QFD was born in this environment as a method or concept for new product development under the umbrella of Total Quality Control (Yoji90). QFD is a process for organising cross-functional thinking to deliver results that have desirable characteristics. It is an ideal method for 1) identifying and organizing needs that may come from the customer, 2) weighting needs, 3) developing and organizing "Quality Characteristics" and metrics, 4) weighting metrics and setting target values, 5) mapping metrics onto systems, subsystems, components and processes, 6) flow down of priorities, weights, and target values. QFD is about planning and problem prevention, not problem solving. It provides a systematic approach to identify which requirements are a priority, for whom, when to implement them, and why. QFD involves the refinement of requirements using matrices and charts based on priorities defined at group level. It consists of many tools, however in this study we focus on the House of Quality (HoQ).

House of Quality (HoQ)

The House of Quality (HoQ) matrix is the most recognized and widely used format of the QFD family of methods. It translates customer requirements, based on marketing research and benchmarking data, into an appropriate number of engineering targets to be met by a new product design. As depicted in Figure 22, the HoQ allows one to fill in a matrix that maps functional requirements to the quality attributes, referred to as Critical to Quality (CTQ). Normally designers and software engineers want the functional requirements to conform to / operate at the level of these CTQs. As the designers move from one level of the House of Quality to the next, the non-functional requirements become more and more measurable, thus more observable and controllable. These measurable requirements are utilised as directives that drives the model and design of the software. In DFSS these controllable attributes are called the Xs of the system upon which the Ys (output) depends.

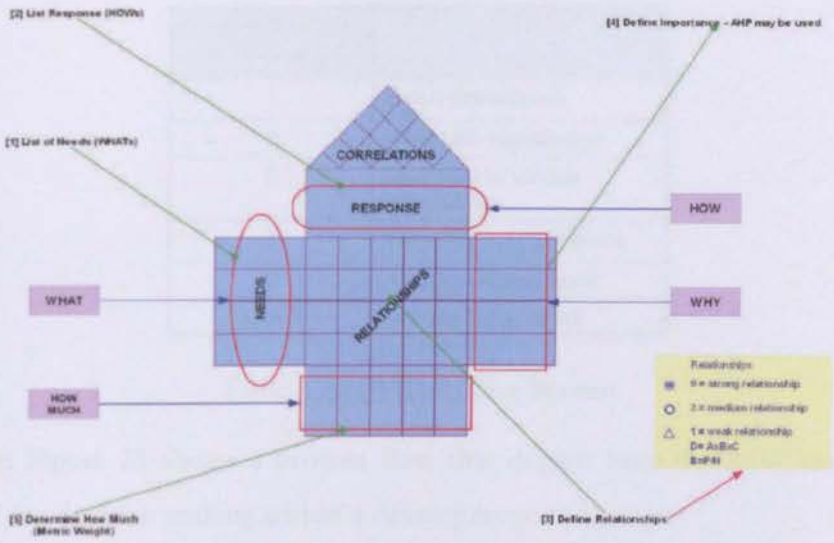

Figure 22 The House of Quality (HoQ)

3.2.3.3 Analytical Hierarchy Process (AHP)

Analytical Hierarchy Process (AHP) was developed by Thomas Saaty, (saat80), and applied to software engineering, amongst others, by Joachim Karlsson and Kevin Ryan in 1997 (Karl96). AHP is a method for decision making in situations where multiple objectives are present. This method employs a pair-wise matrix to calculate the relative value and costs requirements. By using AHP, the requirements engineer can also confirm the consistency of the result. AHP can prevent subjective judgment errors and increase the likelihood that the results are reliable. It is a method for structuring the decision making process into levels that corresponds to **one's understanding** of the problem. AHP is used in community decision making, 1) to elicit preferences for certain objectives comparatively to other objectives and 2) to give the best (or several best), solution(s) from a range of potential solutions. AHP is a multi-attribute decision process that tolerates uncertainty and inconsistency in data and judgments. It employs judgements of the variables that make up a decision and considers the data from those variables. The Table 1 shows the Saaty scale of prioritisation.

Intensity of relative importance (R)	Definition
1	Equal importance
3	Moderate importance
5	Essential or strong importance
7	Very strong importance
9	Absolute importance
2,4,6,8	Intermediate values

Table 1 AHP Weighting System

The model in Figure 23 shows a process flow that depicts how the AHP methodology is implemented for decision making within a development or a project.

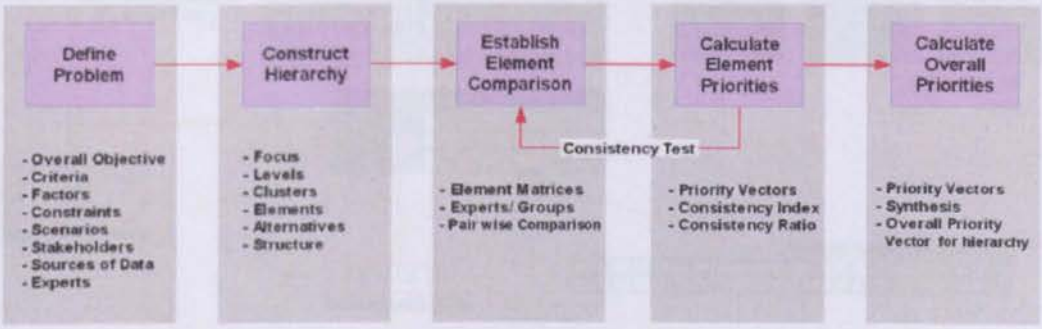

Figure 23 AHP Process Flow

The AHP represents a weight matrix that maps attributes against attributes. Figure 23 depicts the stepwise process of the AHP tool which is based on subjective input from its user and provides a Consistency Index (CI) and Consistency Ratio (CR).

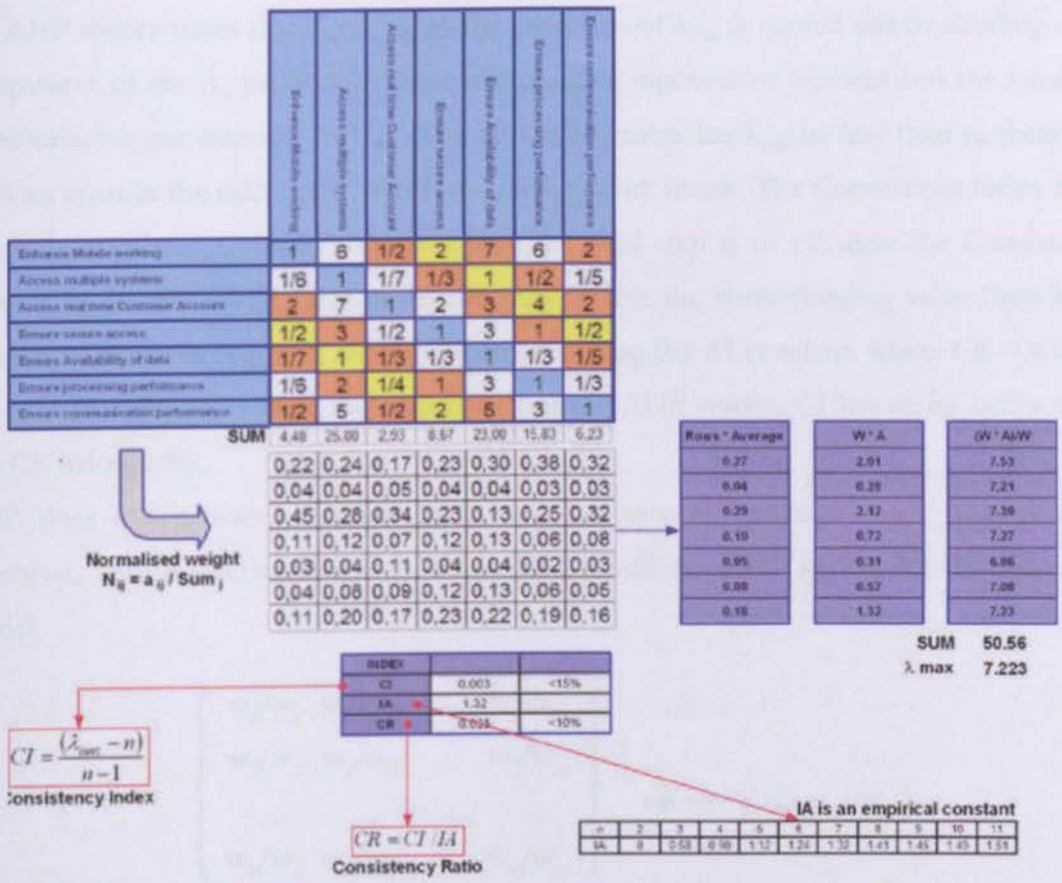

Figure 24 The Analytical Hierarchy Process (AHP)

Consider n elements to be compared, $Q_1 \ldots Q_n$ and denote the relative significance (weights) of Q_i with respect to Q_j by a_{ij} and form a square matrix $A=(a_{ij})$ of order n with the constraints that $a_{ij} = 1/a_{ji}$, for $i \neq j$, and $a_{ii} = 1$, all i. Such a matrix is said to be a reciprocal matrix.

The weights are consistent when the matrix is transitive, that is $a_{ik} = a_{ij}a_{jk}$ for all i, j, and k. Such a matrix exists if a_{ij} are calculated from exactly measured data. Then find a vector ω of order n such that $A_\omega = \lambda_\omega$. For such a matrix, ω is said to be an eigenvector (of order n) and λ is an Eigen value; to resolve a consistent matrix, $\lambda = n$.

There are several methods for calculating the eigenvector. One way is to multiply the entries in each row of the matrix and then taking the n^{th} root of that product provides an approximation to the correct answer (saat80). The n^{th} roots are summed and the sum is used to normalise the eigenvector elements to add to 1.00. λ_{max} is calculated so as to lead to the Consistency Index and the Consistency Ratio. We multiply on the values of the judgements by the eigenvector to obtain a new vector. This vector of n elements is the product A_ω and

the AHP theory states that $A_\omega=\lambda_{max\omega}$ so the estimates of λ_{max} is carried out by dividing each component of the A_ω product by the corresponding eigenvector element and the mean of these values is the estimate for λ_{max}. Should the estimates for λ_{max} be less than n, there has been an error in the calculation, which is a useful sanity check. The Consistency Index for a matrix is calculated from $(\lambda_{max}-n)/(n-1)$ and, the final step is to calculate the Consistency Ratio (CR) for the set of judgements using the CI for the corresponding value from large samples of matrices of purely random judgments using the AI constant, where CR = CI/AI (saat80). In order to validate the consistency of the AHP results, CI has to be below 15% and CR below 10%.

AHP does not provide an exact calculation but instead produces some tangibility to otherwise, very subjective attributes. In an ideal world the AHP matrix fits the following model.

$$\mathbf{A^*} = \begin{bmatrix} w_1/w_1 & w_1/w_2 & \dots & w_1/w_n \\ w_2/w_1 & w_2/w_2 & \dots & w_2/w_n \\ \dots & \dots & \dots & \dots \\ w_n/w_1 & w_n/w_2 & \dots & w_n/w_n \end{bmatrix} \rightarrow (A^* \times W) = nW$$

However, humans are inconsistent and the A* remains unknown. We can only model an A which is close to A*, the closer the better. This implies (A X W) = λ max (W). Hence if A is

$$CI = \frac{(\lambda_{max} - n)}{(n-1)} \rightarrow 0$$

close to A* then the consistency index is very small, close to zero. Generally (λ max > n).

3.2.4 Modelling Concepts and Techniques

Unlike many engineering fields, software engineering is a particular discipline where the work is mostly done on **models** and rarely on real tangible objects. According to Shaw, (Shaw90), Software engineering is not yet a true engineering discipline but it has the potential to become one. However, the fact that software engineers' work mainly with models and a certain limited perception of reality, Shaw believes that the success in software engineering lies in the solid interaction between science and engineering. In 1976, Barry Boehm (Boeh76)

proposed the definition of the term Software Engineering as the practical application of scientific knowledge in the design and construction of computer programs and the associated documentation required to develop, operate, and maintain them. This definition is consistent with traditional definitions of engineering, although Boehm noted the shortage of scientific knowledge to apply.

On one hand, science brings the discipline and practice of experiments, i.e. the ability to observe a phenomenon in the real world, build a model of the phenomenon, exercise (simulate or prototype) the model and **induce** facts about the phenomenon by checking if the model behaves in a similar way to the phenomenon. In this situation, the specifications of the phenomenon might **not** be known **upfront** but induced after the knowledge about the phenomenon is gathered from the model. These specifications or requirement are known ***a posteriori***.

On the other hand, engineering is steered towards observing a phenomenon in reality, **deducing** facts about the phenomenon, build concrete blocks; structures (moulds) or clones based on the deduced facts and **reuse** these moulds to build a system that *mimics* the phenomenon in reality. In this situation, the specifications of the phenomenon are known **upfront**, i.e. deduced before even constructing any models, whilst observing the phenomenon. The process of specifying facts about the phenomenon is rarely a learning process, and requirements are known **a priori**.

The scientific approach is based on **inductive** modelling and the engineering approach is based on **deductive** modelling. Usually in software engineering we are very familiar with the deductive modelling approach, exploiting modelling paradigm such as UML, ERD, and DFD that are well established in the field. However, the uses of inductive modelling techniques are less familiar in business critical software engineering, but applied extensively in safety critical software engineering and academia. Typically, inductive modelling techniques are experiments carried out on prototypes, or simulation of dynamic models which are based on mathematical (formal methods), statistical and probabilistic models. The quality of the final product lies in the modelling power and the techniques used to express the problem. As mentioned earlier, we believe that the power of the modelling lies in the blending of the inductive and deductive modelling techniques.

The rationale of integrating inductive modelling techniques within the domain of our study is due to the elements of non-determinism, emergent behaviours, communicational dynamics which are those parts of the problem that **cannot** be known or abstracted upfront i.e. **a priori**. These elements differs from those parts of the problem that **can** be abstracted **a priori** based on experience and domain knowledge, which are normally deduced and translated into structures or models (moulds) i.e. using deductive modelling techniques.

Inductive modelling techniques require a different approach of addressing the problem attributes. In these circumstances, we tend to believe that the requirements are **false upfront**, and the objective is to validate these requirements against predefined CTQs. To do so, we build models to mimic the functionalities of the suggested requirements and run the models (dynamically) to check if the models conform to the expected output and agreed CTQs. The modelling tools are dynamic in nature, and very often they provide simulation engines that allow system designers to run and exercise the designs, to perform model validation and verification. Through several simulation runs, the models are modified, adjusted and reinforce until they match, to certain level of confidence, the CTQs.

Leveson (Leve95), stresses in that the majority of accidents in which software is involved can be traced to requirements flaws, it is of particular importance to develop complete and unambiguous requirement specifications for software, in which, case, inductive modelling techniques provide the necessary methods and discipline to maximise the reduction of ambiguity. Parts of our class of problems require the use of inductive modelling techniques, hence the following sections provide an evaluation of the various dynamic and simulation tools we have investigated during the course of this study.

3.2.4.1 Coloured Petri Nets (CPN)

The foundation of Petri Nets was laid by the pioneering work of Carl Adam Petri and his colleagues in the early 1960's (Pet62). Since then, an enormous amount of material has been developed, presented and published in journals, books, workshops and conferences. In 1981, Peterson explains the Petri nets methods as mathematical formalism that is intended to be used for modelling, simulation and analysis of different types of systems (Pete81). In the domain of computer science and recently in software engineering, Petri nets are used for modelling a large number of either hardware, software systems and essentially various

applications in computer networks. Moreover Petri nets have special advantages over much other formalism, due to their graphical notation, which reduces Petri nets learning time and simplifies their use through logical abstractions. (Jeff91).

Petri Nets are modelling methodology for the description of concurrent and distributed system behaviour (Jen97, Kris98). A number of versions of this graphical modelling language exist, which find widespread application in verification, simulation and performance analysis of distributed and parallel systems that also includes communication protocols. A classical Petri Net is a collection of places, arcs and tokens (resources), and these tokens are the marker of a place. Each arc connects a place to a transition or vice versa. Places can be considered as conditions on the system control states and transitions as actions. When a condition is verified that implies one or more tokens fill all input places, a transition can trigger and carry the enabling token to the places of destination. When more than one transitions are simultaneously enabled which represent the non-deterministic behaviours, in the validation phases, all the possible execution sequences are explored. But during the simulation phase, the execution step has to be chosen randomly or interactively. Petri Nets includes many variants, from the simplest one (Boolean/Integer token) to the high-level token, where tokens are structured values that, by traversing the net, enable an effective communication between parts of the system. For each class there is a set of software tools.

Tokens carry, as a timestamp, the deterministic or randomly distributed temporal length of the transition they enable. A time consuming transition, then, increments the delay associated to the timed token it processes, which can be consumed by the following transitions only when such delay has elapsed. Communication is represented by token exchange between different parts of the Petri net. A Token "*reception*" can be conditioned by guard conditions appended on the arcs entering the transition or on the transition itself. Based on the current value of the token, which is copied in a locally bound variable, a transition is blocked or enabled and different values for the output tokens can be delivered to the final states.

Jensen (Jen97), clearly illustrates the relation between high-level and ordinary Petri Net and stated that the relationship is analogous to the relation between high-level programming languages and assembly code: the modelling capabilities should be the same, but the superior abstraction of the high-level specification simplifies the design phase. A powerful dialect of

high-level Petri Net are the Coloured Petri Nets (CPN), a methodology that employs Petri Net features to model parallel behaviour and high-level programming languages to define data, functions, and computation on data.

Many freeware and commercial tools to handle modelling, simulation, and validation of low and high-level Petri Nets have been developed (PNT06). In the Telecoms, CPN tool is one of the most elaborate and successful. Of particular interest to this study is the application of CPN to the mobile technology, (Lore01), which illustrates how feature interaction patterns of a mobile phone family can be modelled in CPN and analysed in both time and space. CPN supports hierarchical description by means of transition refinements, model checking for static and dynamic properties, formal analysis methods (*"places"* and *transitions* invariance verification) and simulation (automatic as well as interactive) with feedback information and several debugging options. For the analysis of complex models, a highly efficient simulation engine has recently been developed, which dramatically speeds up automatic simulation runs.

Despite the fact that a modelling idea is sometime hard to create (or model), Petri Net, with its appealing graphical notation is a powerful modelling language to describe and investigate communication and resource-sharing processes. By combining basic formalism and the effectiveness of the programming language for computational details and communication aspects, CPN enable the creation of compact but accurate system models taking into account temporal issues. Despite the richness of the graphical notation, validation and performance analysis is pertinent due to the availability of powerful tools. However, CPN tool requires some effort to gain an acceptable knowledge of the functional language (Standard ML) supported by the tool, which is not a typical implementation language (Ullm98).

3.2.4.2 Simple Promela Interpreter (SPIN)

In the early 80's, Simple ProMeLa Interpreter, (SPIN), was designed for protocol verification systems based on on-the-fly reachability analysis (Holz81, Holz85, Holz88) .SPIN is a widely used tool for specification and validation of communication models that supports asynchronous process systems (Holz91, Holz92). It has been developed with the objective of thoroughly supporting protocol engineering, thus incorporates a strong formal basis and supports efficient model-checking, i.e. validation of consistency in requirements, invariant

assertions, and temporal properties expressed in Linear Temporal Logic. SPIN uses a C-like specification notation (ProMeLa — Process Meta Language) which increases its applicability in the first stages of the design and makes code implementation a rather mechanical task of quantitative detailing.

ProMeLa comprises the constructs for data manipulation and communication between processes. The system's basic components are the processes, whose internal behaviour is described as a set of possible transitions (gathered within the "*if... fi*" construct). The firing of a transition leads to termination (execution states labelled with *end* :) or non-terminating sequences of actions (when the transitions are part of a loop "*do... od*" or a *goto* operator points to a new control state). The triggering conditions guarding each branch are message receiving and sending actions, which are true if executable, 1) Boolean expressions on local or global variables or on the status of visible channels and 2) timeout events, executed only when no other transition is enabled and the system is stalled. All starting conditions inside "*do... od*" and "*if... fi*" constructs are introduced by operator ":". All other actions in process behaviour are delimited by operator ";". Processes can communicate on channels, which are modelled as finite-length queues. The default communication is therefore asynchronous, but can also be synchronous when relying on channels of zero length. On these rendezvous channels, sending and receiving actions must be matched to be executed (they are interpreted as an atomic action), otherwise a deadlock is detected.

Channels deliver structured messages: sending actions are specified with the expression *ch! expr1, expr2,...* or equivalently with *ch!expr1 (expr2,...)*, where the first field is usually the message identifier. The reception of a message can be interpreted as data transfer (value-passing) — *ch?var1,var2,...*, when the values received on channel *ch* overwrite the local variables used by the receiving process. Information exchange can also be performed by writing and reading global variables, here interpreted as shared channels. This approach can help in modelling multicast transmission.

When given a ProMeLa specification file, SPIN performs simulation and validation on it. Preliminary analysis is carried out with random or interactive simulations. For a more detailed inspection, the validator checks for invalid end-states (deadlocks); non-progress cycles; violation of temporal claims, and so on, tracing back the shortest execution sequences to the incriminated states. When the system is too huge to be handled due to the bounds in

available memory, the validation is carried out on a randomly selected subset of the whole state-space (sub-optimal solution). A slicing routine has been added to extract, from complex protocol descriptions, partial specifications related to the properties of interest. These concepts are powerful enough to cover almost all the correctness and consistence requirements that can be imposed on prototypes.

SPIN is the most popular formalism among freeware Formal Method tools for the specification and verification of communication protocols. Theoretical advancements and practical experiences are continuously reported in the proceedings of the International SPIN Workshops. The power of SPIN as a verifier, together with its free availability for commercial purposes, justifies its frequent usage in the verification of large models. SPIN has proved to be very useful in the simulation and validation of system specifications. Modelling protocols from technical specification documents is rather clear-cut and results in compact specifications, but specifying a complete protocol stack is not yet possible. But, without relying on rapid-prototyping, the software implementation of protocols is effortlessly obtained from the formal model due to the similarities between C and ProMeLa.

However, ProMeLa has no clear separation between architectural and functional aspects. Due to the lack of hierarchical structuring facilities, it is not possible to point out the interdependencies between processes without looking at their behaviour. As far as this application is concerned, an underlying computational environment, such as inclusion of C code fragments, is indispensable.

3.2.4.3 Unified Modelling Language (UML)

UML (Unified Modelling Language) is the most used graphical modelling notation for object-oriented software systems, whose standardisation is co-coordinated by the Object Management Group (OMG) (UML97). UML specifications are collections of diagrams, where each of them supports a specific phase of the development process by capturing a particular abstraction of the system behaviour (analysis of the requirements, architectural and functional design, HW/SW partition, etc.).

The following diagrams are useful for protocol specification:

- Sequence diagram

- Class diagram, a graph of abstract elements (classes) connected by static relationships: hereditary, aggregation, association, and so on.
- UML's State-chart diagram is a graph that statistically illustrates the dynamic behaviour of classes representing event-driven processes. Its UML notation is derived from state charts, a powerful Formal Description Techniques (FDT) that extends the finite state machine models with parallel decomposition of states and transversal communication between different hierarchical levels of the specification ().
- Object diagram, which collects the class instances (objects) implementing the actual system.

In UML, processes are modelled with classes, input signals with public methods, messages, and information elements with methods' parameters. Classes communicate through the invocation of other classes' methods, which triggers an internal reaction and, possibly, a response. In order to support syntactic and semantic verification, UML includes OCL (Object Constraint Language), a formal language to specify invariant conditions and well-formed rules on the system model. As for the extension of the modelling language to real-time systems (Real-Time UML), a sound mathematical foundation is still under investigation. There are dozens of commercial and freeware tools for UML, ranging from simple graphical editors to complete CASE packages.

UML does not intrinsically support a formal method of verification, yet several investigations have been carried out to attach the component of simulation, model checking and formal verification into the platform of UML. One of the distributions is called vUML, which is a transformational tool able to convert UML models into PROMELA descriptions, which can then be processed by SPIN and this technology is developed at the Åbo University (Finland), under the direction of Professor Johan Lilius (Lilu00). vUML automatically verifies UML models which are programmed in Python, an interpreted OO programming language (Asch99). The models to be checked by vUML are described as Python programs as well, using the UML meta-model implemented by the tool. vUML verifies models where the behaviour of the objects is described using UML State charts diagrams. It supports concurrent and distributed models containing active objects and synchronous and asynchronous communication between objects. The tool uses SPIN model checker to perform the verification, but the user does not have to know how to use SPIN or

the PROMELA language. If an error is found during the verification, the tool creates a UML sequence diagram showing how to reproduce the error in the UML model.

UMLAUT (Unified Modelling Language All pUrposes Transformer) is another example of a general transformational framework for UML models. It has been developed and is maintained within the PAMPA project (*Modèles et Outils pour la Programmation des Architectures Parallèles Réparties*) under the direction of Jean-Marc Jézéquel at IRISA (*Institut de Recherche en Informatique et Systèmes Aléatoires, Rennes, France*) (Ho99). UMLAUT is not directly dedicated to verification but rather to rewriting of UML models for various purposes (among others, the generation of verification models).

3.2.4.4 Specification and Description Language (SDL)

The language SDL has been developed by the Comité Consulatif International Télégraphique et Téléphonique (CCITT) in 1976 and revised in 1980, 1984 and 1988 for use in the design of Telecoms systems. SDL is a formal notation, a high-level general-purpose description language for event-driven, real-time and communicating systems; Telecoms systems and protocols are one of its main application fields (CCITT). The effectiveness and the graphical format of SDL have won it a widespread popularity in both the academic and industrial sectors. SDL is based on Finite State Machines (FSM), a standard technique for studying reactive systems. The system described by an SDL specification is actually an Extending Communicating Finite State Machine (ECFSM), because it consists of a set of concurrent processes, extended with variables and data space, which communicate by exchanging control signals or structured on finite-length asynchronous channels. In system specification, structural and functional aspects are distinguished. The system architecture deploys the building blocks and the channels connecting them and by declaring the signals exchanged on each communication path. The behaviour of each basic block is detailed by editing the internal processes with respect to their required operations. The association among blocks and processes is accomplished in the block diagram, which contains the declaration of the processes (one or more for each block) and the routes connecting each process to other internal processes (if any) or to the block's interface.

SDL supports non-determinism (i.e. **ANY** concept, representing an alternative of equally possible actions), which can be useful to describe uncertainty in the environment and to validate accordingly all possible reactions of the system. Processes can handle local variables and parameters of different data types. The data type concept in SDL was originally based on axiomatic definitions of Abstract Data Types (ADT), defined by the set of possible values and the operations that can be executed with them. With ADT it is possible to characterise types through inheritance, thus avoiding redundant redefinition of common operators, and to define parametric data types, i.e., type generators, used for example to define arrays of elements of any type, whose index can be of any type supporting internal ordering. Due to the complexity of ADT, a set of predefined data types and type generators are provided.

In SDL, communication on the bi-directional routes that connect processes is always asynchronous: that is, each process has its own input queue, where signals coming from the environment or from another process are buffered and sequentially processed. Synchronous communication is provided by the remote-procedure call instruction. SDL also supports non-ordered signal reception by means of the SAVE operator, which postpones the consumption of a specified signal to the following transition.

Several software packages have been developed to handle formal specifications of systems in SDL, amongst the most popular is JADE (Leue96, Henk96, Henk97). Yet many studies on these packaged are centred on the problem of inefficiency in the code generation practice of translating formal languages into codes, which the authors mainly claim is mostly due to the fact that the specification model is not the most efficient to be used in the implementation phase but it is the one that guarantees full coherence between both. However, JADE is the most promising as a public domain tool for specification written in Java, for which code generation and optimisation have been developed. Another SDL software package is SITE, which is an open development environment, support compilation of SDL and ASN.1 to target languages Java and C++. Unfortunately, these tools are not yet mature enough to adequately support system development with SDL and offer no significant alternative to commercial software.

For many years, SDL has been successfully applied to system analysis and design in many application domains. Although SDL provides timers, performance evaluation and analysis of temporal properties are poorly supported. Tools for performance analysis of queuing

systems specified in SDL have been proposed, but are not yet sufficiently mature for professional use. As far as the modelling phase is concerned, the modular approach and the clear distinction between structure and behaviour are very useful features in describing Open Systems Interconnection (OSI) like protocol architectures. In addition to this, the translation of the informal textual specifications issued by ETSI[1] and 3GPP, for example, to SDL diagrams is a very straightforward task because the style adopted complies with SDL concepts and operators.

3.2.5 AHP Evaluation of Modelling and Simulation Tools

We are now faced with several dynamic modelling techniques and we are required to evaluate each of them against specific CTQs. These CTQs are the quality attributes, which we believe are necessary for modelling and simulating the properties and characteristics of distributed messaging systems. During the evaluation phase, we spelt out that the following CTQs are required properties of the dynamic modelling tools; 1) simulation engine, 2) temporal properties, 3) abstraction, 4) user interface (ease to use), 5) concurrency and 6) formal verification. Since the evaluation process also include subjective assessment using domain knowledge and expert groups, we planned to apply the AHP to prevent subjective judgment errors and increase the likelihood that the results are reliable.

Therefore, the exercise in Figure 25, illustrates the AHP evaluation techniques. The first task is to evaluate the quality attributes that are the most critical to the modelling phase of the study and assess them against each other. This provides a weight per CTQ which is represented by the column CTQs.

[1] European Telecommunications Standards Institute

Factor	Simulation	Temporal Prop	Abstraction	User Interface	Concurrency	Formal Verification	GEOMEAN	Normarized Weight	CTQs
Simulation	1.00	2.00	2.00	3.00	2.00	2.00	1.9063686	0.2841646	28.41646
Temporal Prop	0.50	1.00	2.00	3.00	0.50	2.00	1.200937	0.17901248	17.901248
Abstraction	0.50	0.50	1.00	2.00	0.50	0.50	0.7071068	0.10540182	10.5401818
User Interface	0.50	0.33	0.50	1.00	0.33	0.50	0.4902805	0.07308154	7.30815391
Concurrency	0.50	2.00	2.00	3.00	1.00	2.00	1.5130857	0.22554159	22.5541592
Formal Verification	0.50	0.50	2.00	2.00	0.50	1.00	0.8908987	0.13279797	13.279797
						Total	6.7086772	1	

0.2841646	0.35802496	0.21080364	0.21924462	0.451083185	0.26559594	6.2953547
0.1420823	0.17901248	0.21080364	0.21924462	0.112770796	0.26559594	6.3096705
0.1420823	0.08950624	0.10540182	0.14616308	0.112770796	0.06639898	6.2837931
0.1420823	0.05967083	0.05270091	0.07308154	0.075180531	0.06639898	6.4190642
0.1420823	0.35802496	0.21080364	0.21924462	0.225541592	0.26559594	6.3016893
0.1420823	0.08950624	0.21080364	0.14616308	0.112770796	0.13279797	6.2811504
						37.890722

λmax =	6.31512037	
C.I. =	0.06302407	< 0.16
C.R. =	0.05062587	< 0.15

Figure 25 AHP Evaluation of CTQs for Simulators

In this case we observe that simulation, as a quality attribute, is the strongest CTQs, hence the next phase of the AHP exercise is to evaluate each tool against the six Quality attributes, which typically is a benchmarking exercise. The diagram in Figure 26, shows the assessment of the each formal verification techniques against each individual CTQ.

The results of the AHP reveal that the specifications of CPN best fit our quality requirement of the formal verification techniques. Among the notations devoted to system specification, SDL has achieved widespread success for its friendly graphical notation, its conformance to the idiom adopted by standardisation institutes, and its support for other popular notations such as ASN.1, MSC, and TTCN. SPIN outperforms SDL-based commercial tools because it can check, besides static and dynamic properties, the consistency of the system's temporal behaviour in all execution sequences. This makes SPIN the most successful freeware tool for large-system validation by both academic and industrial users. As a formal verifier, SPIN is very robust. However, SPIN does not provide its user with model abstractions and a friendly environment to develop real-life business critical projects.

Figure 26 AHP Benchmark of Simulation Tools

Petri nets, on the contrary, adopt a friendly (although unconventional) graphical notation to address several aspects of the development process: formal modelling, model-checking, and efficient simulation of high-level prototypes. Timed automata analyse temporal properties of non-deterministic timed systems, verifying the correctness of the specification and calculating temporal hard bounds on the system performance. This kind of check is indispensable for safety-critical real-time. According to the AHP analysis, CPN proved to be a good candidate for dynamic modelling.

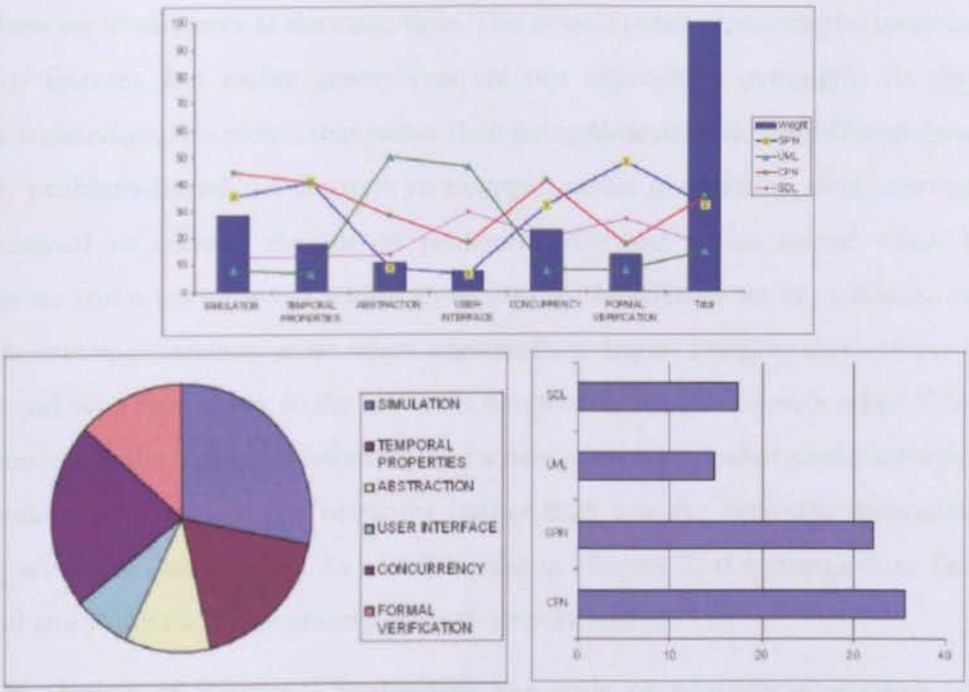

Figure 27 AHP Result of Benchmark for Simulation Tools

CPN allows a user to simulate different scenarios of a system, through automatic or interactive simulations. Interactive simulation gives the user control over the procedures and an analogy is single stepping while debugging some codes using a high level compiler. The user can step in and out of the system while the simulation runs. CPN also allows the user to investigate the different state of transition during simulation and as a result of which the user can enable or disable any transition. In brief, interactive simulation is a powerful feature of CPN that tolerates logical walkthroughs a model.

On the other hand, automatic simulation runs on pre defined parameters without human interaction which is similar to a program execution. CPN offers formal verification known as state space analysis and invariant analysis. It provides proofs in mathematical sense that specifies certain behavioural properties of a system.

3.3 Implementation and Technology

The evolution of network technology does not happen in a linear fashion (Jini01). Network technologies exist in generations, as one generation is born, a few years later the next generation emerge. The earlier generations are still in operation for quite a while, and several

generations are in existence at the same time. The newest generations may be more advanced with new features, but earlier generations are not superseded overnight. As applied to network technology, this means that rather than being **time-based**, the different generations are really **problem-based**. To illustrate an example, earlier generations, *client/server* and *n-tier*, were designed to address the set of problems that had to be solved when building client/server and n-tier networks within an enterprise. A different set of problems, called the *network-to-network* generation, arose when organisations began bringing applications into the Internet and Web framework so the networks could offer services to each other. With the IP convergence and the Telecoms boom, we see a new generation, called *participant-to-participant*, as the individual entities in the networks (rather than just the networks themselves) start offering services to each other. As we addressed in chapter 2, the change is in fact rather profound and complex encompassing general network issues:

- **Evolution of System - Scalability**: the scale of possible interactions increases dramatically if the environment enables every participant to either provide or consume a service to or from every other participant.
- **Quality of Service**: The rate of message exchange and the complexity inherent to each message increases rapidly if services can be offered and consumed by all of the participants of the networks, either due to change in the network or failure in the infrastructures.
- **Communication Model**: The complexity of interactions increases together with the options to provide or consume services from entire networks, individual participants, or both.

. We classified these issues into three distinct areas:

- **Network Technology:** the type of network technology used: Ethernet, Myrinet, InfiniBand.
- **Network Operation Environment:** "*how does the network technology communicate with the application*", i.e. the transport protocols.
- **Application Protocol:** "*How do software applications interface with the network technology?*"

The following sections describe the networking implementation adopted to pursue the objective of designing a distributed messaging system. We start from the lowest level of the (network technologies) and move up to the application protocols.

3.3.1 Network Technologies

3.3.1.1 The InfiniBand Network Model

The InfiniBand architecture connects servers and shared I/O systems. By moving I/O out of servers onto a switched network, InfiniBand technology results in what has been termed "deconstruction of servers" which is anticipated to be a foundational design principle of future data centres (Fut02). The architecture covers all layers of the standard, including Physical, Link, Network, Transport, and Management. The InfiniBand architecture is a product of the InfiniBand Trade Association (IBTA), an industry group of over 200 companies led by Intel, Compaq, Microsoft, Hewlett-Packard, Dell, IBM, and Sun Microsystems. The IBTA was formed in the summer of 1999 as a result of merging the Next Generation I/O (NGIO, led by Intel) and Future I/O (led by Compaq) standards. The IBTA architecture definition work is performed by a series of Working Groups corresponding to layers of the architecture (e.g., Electro-Mechanical, Link, Software, and Management). There are also IBTA working groups focused on Marketing and Compliance and Interoperability. A single InfiniBand network is termed a subnet, which support queuing of messages, including the following

- Host Channel Adapters (HCAs) – interfaces to processor nodes
- Target Channel Adapters (TCAs) – interfaces to I/O nodes (including storage and other networks)
- Switches – interconnection components for routing traffic within a subnet
- Routers – interconnection components for routing traffic across subnets or to non-InfiniBand networks

Each node contains one or more ports that are connected to other ports with point-to-point links. Three speeds of InfiniBand bi-directional interconnection links are supported: x1 (2.5Gb/s), x4 (10Gb/s) and x12 (30Gb/s). The InfiniBand transport layer defines protocols

for both unreliable and reliable messages, as well as Direct Memory Access (DMA) mechanisms. These protocols enable a variety of I/O and Inter Processor Communication (IPC) operations between end nodes. The interface for managing message traffic within a channel adapter is called a Queue Pair (QP). Each QP contains both a *send queue* and a *receive queue* which connect through communication channels. The InfiniBand networks support a number of mechanisms that enable logical control of components and traffic in a subnet. Partitions enable logical isolation of components sharing the same InfiniBand subnet. Virtual Lanes constitute a mechanism for having multiple virtual links share the same physical link. The Virtual Lane 15, which is supported by all InfiniBand ports, is reserved for management layer traffic.

Link Protocol

Figure 28 illustrate the link protocol of the IBA, wherein application data is composed of several messages, size up to 2 GB. The messages can be automatically segmented and assembled into packet that support payload ranging from 256, 512 and 1024 Mb.

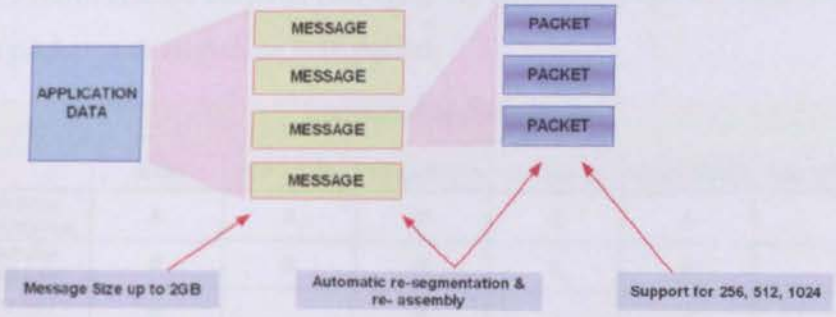

Figure 28 The IB Link Protocol Model

The packets are sent over dedicated buffers of the virtual lanes, as shown in Figure 29.

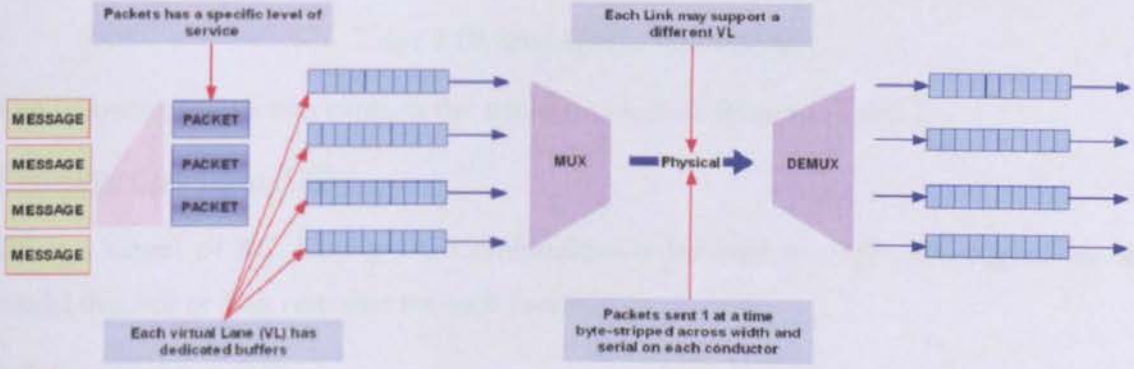

Figure 29 The IB Link Attribute Model

The IBA Network Features

The addressing system of the IBA allows devices to be statically defined by a Global Unique Identifier (GUID) which is a similar notion to Ethernet MAC addresses. All ports in a device have a dynamically assigned Local Identifier (LID) which is a 16-bit address assigned by a subnet manager. Every fabric subnet must have at least one Subnet Manager (SM) and may be implemented anywhere in the fabric and have the following responsibilities:

- Discover subnet topology
- Assign LIDs
- Calculate and program switch forwarding tables
- Monitor changes

Table 2 illustrates the process of the send queue operation.

Reliable Connected (RC)

During RC Communication between two QPs, the responders QP RC logic must respond to each request packet with an Ack or Nak packet.

SEND QUEUE OPERATION

	SEND	RDMA READ	RDMA WRITE	ATOMIC OP	BIND WINDOW	MULTI-CAST
Reliable Connection	X	X	X	X	X	
Reliable Datagram	X	X	X	X	X	
Unreliable Connection	X		X		X	
Unreliable Datagram	X					X
Raw Datagram	X					X

X = ALLOWED OPERATION

Table 2 IB Send Queue Operations

The following sub section explains the attributes of IB as shown in Table 2

Unreliable Connected (UC)

UC is a subset of RC. During UC Communication between two QPs, RC logic does not model the Ack or Nak response for each packet.

Reliable Datagram (RD)

The RD service type permits a QP to send and receive messages to/from a number of remote QPs that have been set up to use the same type of datagram service., where RD requires initial connection establishment and each Connection is needed with an Ack/Nak model.

Unreliable Datagram (UD)

UD service type permits a QP to send and receive messages with any number of remote QPs that have also been set up to use the same type of datagram service. However, UD does not require initial connection establishment, hence there is no need for any connection with Ack/Nak model.

Raw Protocols

The raw protocols are used to send and receive non-IBA packets that are encapsulated within an IB packet, which includes raw IPv6 and Ether Type datagram.

3.3.1.2 The InfiniBand Protocol Stack

Figure 30 shows the upper layer protocols which are typically standardised and implemented in other switched fabric architectures, with the exceptions include IPoIB which is IB specific. The following is a description of the protocols supported by the IB stack as illustrated in Figure 30

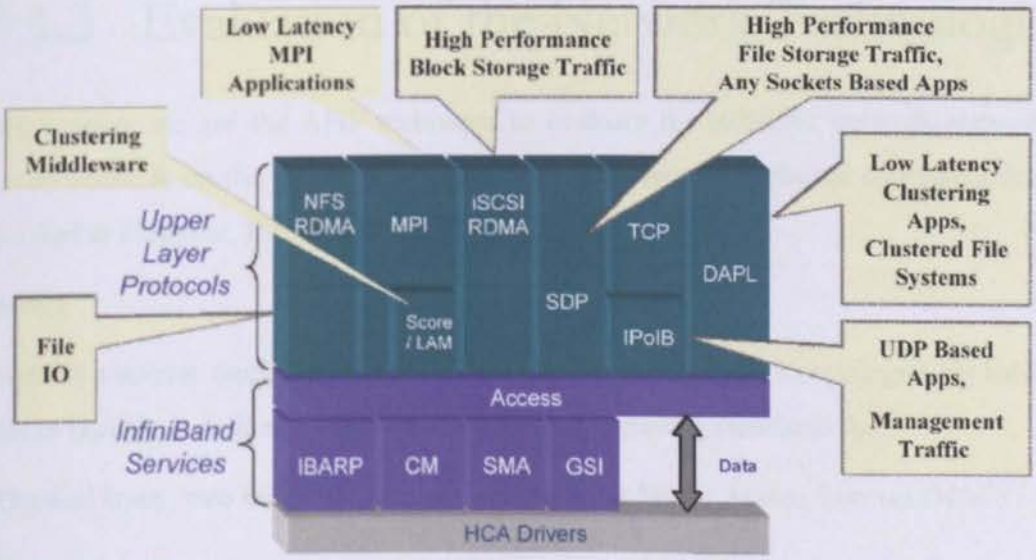

Figure 30 IB Protocol Stack Model

IP over InfiniBand (IPoIB) provides a standard mechanism to encapsulate IP traffic over IB fabric which supports TCP, UDP and Internet Control Message Protocol (ICMP) based applications and traffic.

Socket Direct Protocol (SDP) maps standard socket (Winsock, Berkeley Socket, (BSD), etc) API's to IB architecture, keeping the same API's towards the application in a way that the application still thinks it is working over a TCP/IP based network, but with much greater performance and lower latency (as it bypasses the TCP/IP stack). SDP takes advantage of Asynchronous IO and RDMA and used for IPC between IB nodes and for session based networking with elements outside the IB fabric residing in the LAN/WAN. This efficient connectivity is achieved by using Intelligent Routers that terminate TCP/IP traffic translating it to SDP. SDP works in conjunction with the IP over IB standard, which enables routing and non TCP Traffic (ICMP, UDP) to traverse between the nodes.

Direct Access Provider Library (DAPL) is not a protocol but an industry attempt to define and standardize a transport independent and OS independent set of APIs, that exploit RDMA and other capabilities present in IB. There are two flavours uDAPL and kDAPL (user and kernel implementation modes respectively). DAPL supports Direct Access Transport (DAT) over IB and uses IP addressing and IPoIB for connection establishment. MPI is an application protocol or message-passing, proposed as a standard by a broadly based committee of vendors, implementers, and users.

3.3.1.3 Evaluation of the Network Technologies

In this section, we use the AHP technique to evaluate the different network technologies currently available on the market for the purpose of building distributed messaging systems. We looked at Ethernet, InfiniBand and Myrinet.

Ethernet

Ethernet is a diverse family of frame-based computer networking technologies for local area networks (LAN). It defines a number of wiring and signalling standards for:

- Physical layer - two means of network access at the Media Access Control (MAC)
- Data link layer

- Common addressing format

Ethernet has been standardized as IEEE 802.3. Its star-topology, twisted-pair wiring became the most widespread LAN technology in use from the 1990s to the present, largely replacing competing LAN standards such as Coaxial-cable Ethernet, Token Ring, FDDI, and ARCNET.

Myrinet

Myrinet is a high-speed local area networking system designed by Myricom to be used as interconnects between multiple machines to form computer clusters. The technology is often used directly by programs that are aware about it, thereby bypassing a call into the operating system. Myrinet physically consists of two fibre optic cables, upstream and downstream, connected to the host computers with a single connector. Machines are connected via low-overhead routers and switches, as opposed to connecting one machine directly to another. Myrinet concepts include a number of fault-tolerance features, mostly backed by the switches. These include flow control, error control, and "heartbeat" monitoring on every link.

AHP Evaluation

We carried out an AHP benchmark to decide on the best network technology candidates for the build of a distributed messaging prototype. Figure 31 depicts a sample of the AHP matrices built to evaluate predefined CTQs for each technology. The CTQs are general networking quality of Service parameters extended to meet the requirements of this study. Each of the CTQ was quantified on the AHP based on the typical Telecoms QoS benchmark, (Jung96).

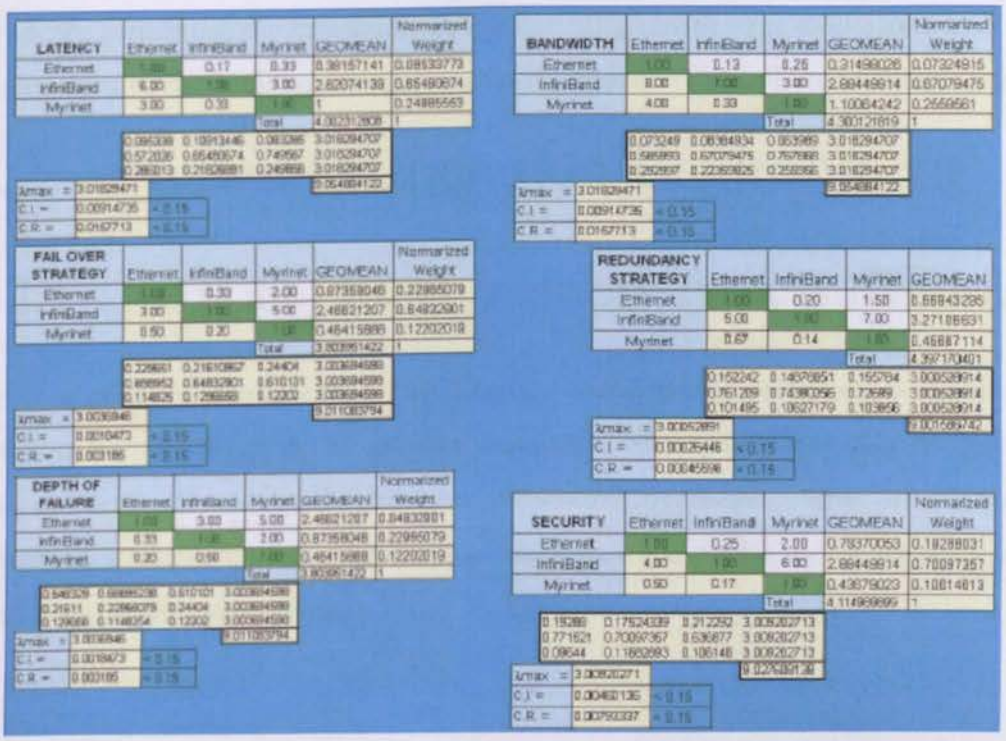

Figure 31 AHP Benchmark of Network Technologies

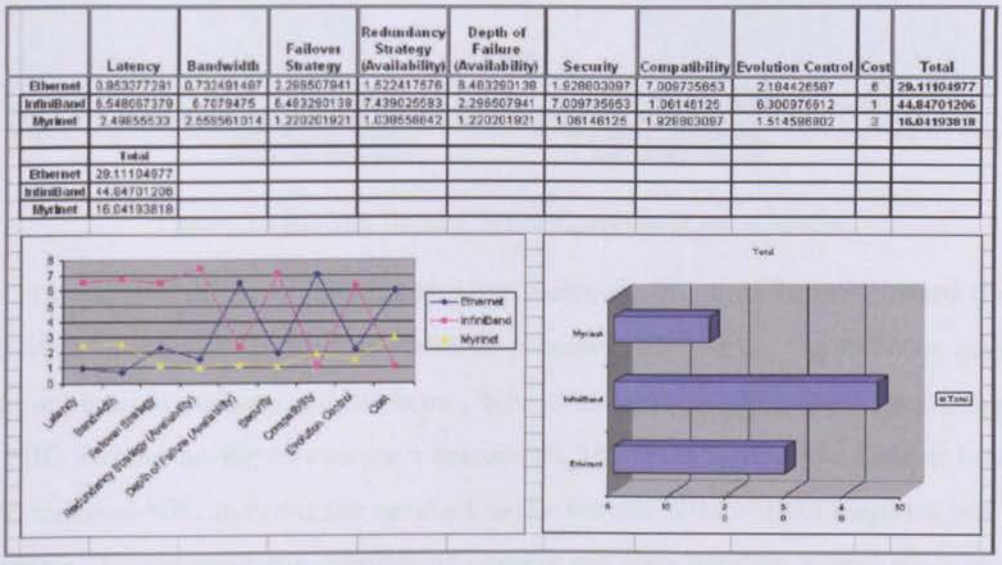

Figure 32 AHP Result of Network Technology Benchmarking

Figure 32 illustrate the result of the AHP benchmarking exercise, providing the necessary visibility and knowledge to discuss on the most appropriate technology. As we assess the three technologies against the CTQs, it became clear that InfiniBand and Ethernet are best candidate technologies to develop a prototype against the very proprietary Myrinet. In

chapter 6, we carried out the simulation and prototyping experiments to illustrate an in depth analysis of the two network technologies within the domain of distributed messaging system.

3.3.2 Network Operation Environment

3.3.2.1 Remote Direct memory Access (RDMA)

RDMA is a network interface card (NIC) feature that lets one computer to directly place information into the memory of another computer. The technology reduces latency by minimizing demands on bandwidth and processing overhead. This is achieved by implementing a reliable transport protocol in hardware on the NIC and by supporting zero-copy networking with kernel bypass.

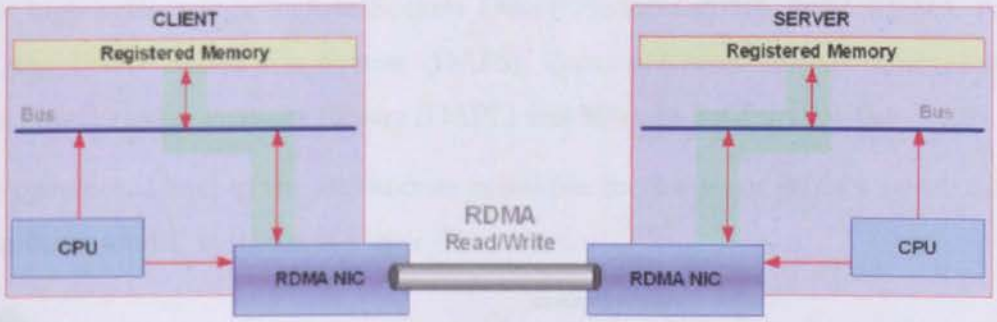

Figure 33 RDMA Read / Write Communication Model

Figure 33 show that the zero-copy networking allows the Network Interface Card (NIC) to transfer data directly to or from application memory, eliminating the need to copy data between application memory and the kernel. Kernel bypass lets applications issue commands to the NIC without having to execute a kernel call. The RDMA request is issued from user space to the local NIC and over the network to the remote NIC without requiring any kernel involvement. This reduces the number of context switches between kernel space and user space while handling network traffic. When an application performs an RDMA read or write request, no data copying is performed. The RDMA request is issued from an application running in user space to the local NIC and then carried over the network to the remote NIC. Request completions might be processed either entirely in user space (by polling a user-level completion queue) or through the kernel in cases where the application wishes to sleep until a completion occurs.

Over RDMA, the remote virtual memory address for the operation is carried within the RDMA message. The remote application is not required to perform any operation other than registering the relevant memory buffer with its local NIC. The CPUs in the remote node are not at all involved in the incoming RDMA operation, and they incur no load. An application can protect its memory from arbitrary access by remote applications through key value use. The application issuing the RDMA operation must specify the correct key for the remote memory region that it is trying to access. The remote application obtains the key value when it registers the memory with its local NIC. The issuing application also must determine the remote memory address and the key for the memory region. The remote application informs the issuing application about the beginning virtual address, size and key of the memory region. It transmits this information using a send operation before the issuing application can start issuing RDMAs to that memory region. Protocols that use RDMA to achieve high performance include Sockets Direct Protocol (SDP), SCSI RDMA Protocol (SRP) and Direct Access File System (DAFS). Communication libraries that use RDMA include Direct Access Provider Library (DAPL) and Message Passing Interface (MPI).

At the operational level of the architecture reside the mechanics of RDMA which follows a basic queuing model, as shown in Figure 34.

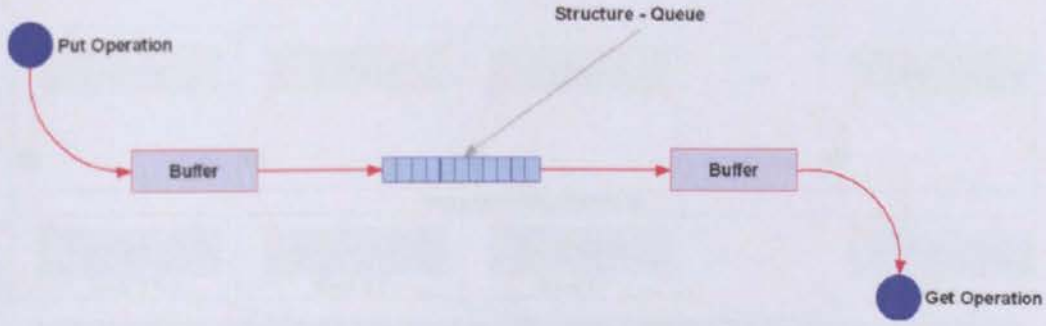

Figure 34 RDMA Queuing Model

3.3.2.2 The RDMA Channel Interface

InfiniBand connectivity over RDMA channel removes latency from data transfer amongst the nodes which occurs through the process of kernel bypass, as shown as Figure 35.

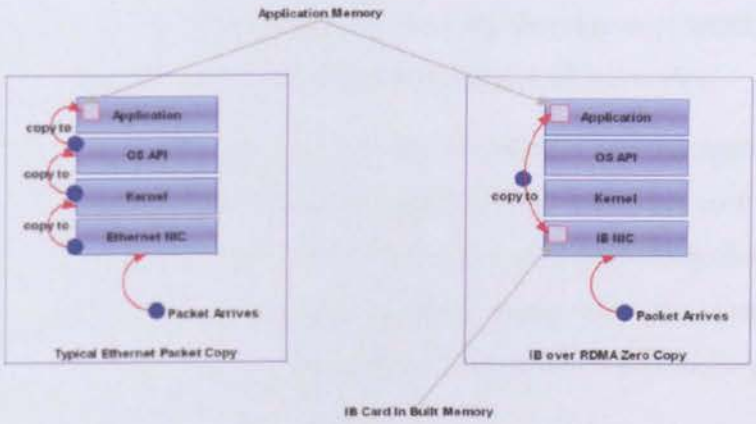

Figure 35 RDMA Zero Copy Model

When an application is designed using RDMA protocols, the Kernel is notified to provide a fix portion of memory to the application giving it the right to manage the memory. As the application is initiated, the memory allocated is mapped to the memory chip built on the InfiniBand NIC. The NIC has direct access to the memory which is now managed by the application rather than the Kernel; hence the overheads (work-extensive copies) are eliminated. Through the fast transport capacity and low latency of InfiniBand, coupled with the zero copy concept of RDMA, the model provide the illusion that remote peers can access each other memory directly, as Figure 36 illustrates.

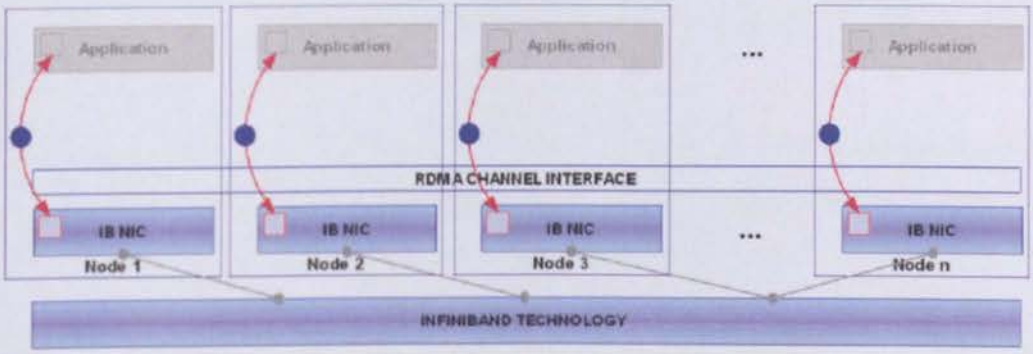

Figure 36 RDMA used in Distributed Shared Memory

RDMA leads us to the concept of Distributed Shared Memory (DSM) wherein, each process "thinks" or has a view of the memory space, hence access a portion of a large shared memory (public segment). The public segment is made up of the contribution of memory segments from each node. As required, the nodes also reserve a private segment of memory for caching mechanism and local operations. In the architectural model of DSM, usually

there no concept of central shared memory, but only the segments which are visible locally and those which are allowed to exported or shared across all the nodes.

The principle of two or more programs sharing a common address space through Remote Direct Memory Access (RDMA) residing on different machines is a new design concept in the Telecoms system infrastructure. Our proposition is to allow dispersed nodes (different physical machines) to share a common address space with the primary objective of collaborating data and knowledge through direct connection to peers memory segment.

3.3.3 Applications Protocols

3.3.3.1 Message Passing Interface

MPI is an application protocol where a complete implementation of the MPI-2 library supports functionalities such as dynamic process management, one-sided communication, and MPI I/O. The MPI-2 process model allows for the creation and cooperative termination of processes after an MPI application has started. It *partly* exhibits the concept of dynamism in distributed communication by providing a mechanism to establish communication between the newly created processes and the existing MPI applications. It also provides a mechanism to establish communication between two existing MPI applications, even if one process did not "start" the other.

MPI library is a standard for writing parallel applications. The library has been implemented for numerous interconnects, including remote memory access abilities. In the study carried out on MPI for Cray T3D (Brig96), MPI has been implemented on a SHMEM interface. The work presented in (Sist02) described an MPI implementation over Sun Fire Link that uses a sender-managed buffer scheme for transporting messages wherein, the sender can choose any buffer at the receiver side for doing remote write.

MPI extended its communication mechanisms in adopting direct memory access to allow one process to specify all communication parameters, both for the sending side and for the receiving side. This mode of communication facilitates the coding of some applications with dynamically changing data access patterns. Each process can compute what data it needs to access or update at other processes. However, in many DMA communication model,

processes may not know which data inside their own memory, need to be accessed or updated by remote processes and these processes may also be unknown. Thus, the transfer parameters are all available only on one side. Regular send/receive communication requires matching operations by sender and receiver. In order to issue the matching operations, an application needs to distribute the transfer parameters. This may require all processes to participate in a time consuming global computation, or to periodically poll for potential communication requests. The use of Remote Direct Memory Access (RDMA) communication mechanisms avoids the need for global computations or explicit polling.

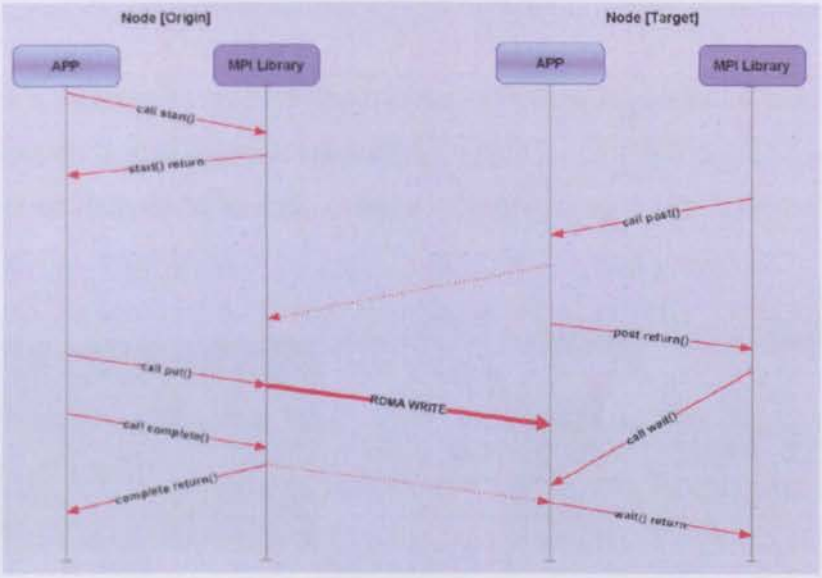

Figure 37 MPI Operational Model

As illustrated in Figure 37, the message-passing communication achieves two effects: communication of data from sender to receiver; and synchronization of sender with receiver. The RDMA design separates these two functions by invoking three communication calls which are MPI PUT (remote write), MPI GET (remote read) and MPI ACCUMULATE (remote update).

The model behind MPI over RDMA channel interface is supported by the work carried out on the evaluation of communication models, which reveals that RDMA calls in MPI is a fast an efficient solution for different designs of implementing remote servers on a cluster connected system (Maui05). Some of the methods evaluated, uses RDMA write for transferring control messages across nodes.

3.3.3.2 MPI Library Comparison Analysis

There are several distributions of MPI libraries and during the course of this research, we evaluated two different libraries. The experiments were carried out by measuring the performance of 4 computational nodes sending data packets of fixed payload to each other (10 bytes). The number of packets sent within a time period (msg / time) is recorded per node for each MPI library, namely, 1) HP MPI from Hewlett – Packard and 2) MPIPro from Verari Systems.

HP MPI

We connected 4 nodes over an InfiniBand fabric and created an MPI world, written using the HP MPI library, to simulate throughput performance across the nodes. Figure 38 shows the throughput of each node, and the average CPU processing time and memory usage of the four nodes.

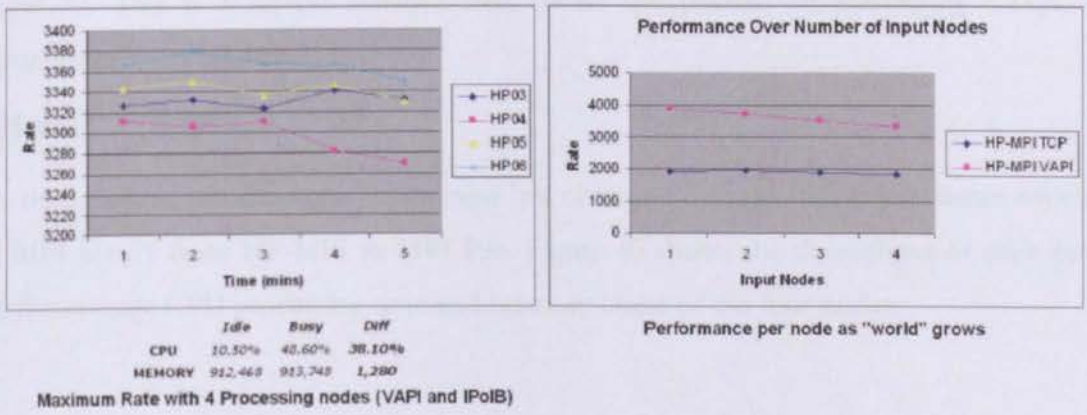

Figure 38 HP MPI Performance Results over IB vs. Ethernet

HP-MPI performed very well but at a cost of processing resources. The throughput for message (sms) exchange was acceptable but the ability to manage fault tolerance aspects was unacceptable for the domain of messaging systems. We were provided with a Beta version of their HP MPI library that addresses the aspect of fault tolerance. In examining the new release with the same simulation parameters, we reproduced the following results, as shown in Figure 39.

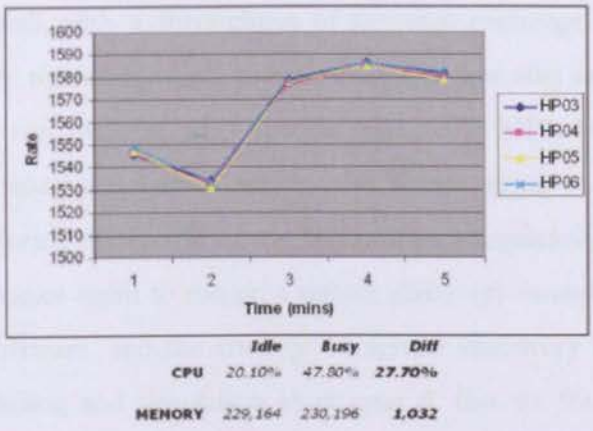

Figure 39 HP MPI with Fault Tolerance

The new HP-MPI Beta release, which incorporated some additional modules called poka yoke, to ensure fail over mechanism should MPI ranks fail and as Figure 39 shows knock-on effect seemed to be a hit on performance when comparing the graphs of Figure 38 and Figure 39. This is a typical instance that shows the process of conflicting CTQs, i.e. performance vs. reliability.

MPI Pro

We, then, carried out the same experiment but changing one simulation parameter which is the MPI library from HP MPI to MPI Pro. Figure 40 shows the throughput of each node, and the average CPU processing time and memory usage of the four nodes.

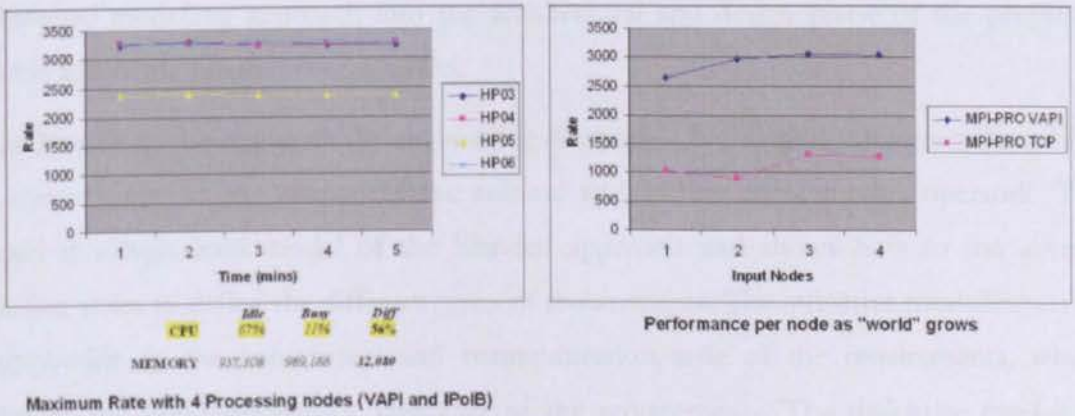

Figure 40 MPIPro Performance Results over IB vs. Ethernet

MPI/Pro performed well, with a throughput of message exchange higher than HP MPI tolerance) yet the ability to manage fault tolerance aspects was also unacceptable. However, MPI/Pro was able to run multiple MPI worlds unlike HP MPI, but we were unable to empower a fault tolerance mechanism within the library since it was not on vendors, roadmap. As a result, during the course of the research we temporarily resolved the problem by enabling a fault tolerance agent to run on a service discovery model based on the Red Hat cluster management software, and the concept of service discovery is thoroughly assessed through dynamic modelling and simulation in chapter 4. But we found that MPI Pro is a very clean implementation of the MPI standard, in terms of performance, RDMA communication model and programming.

3.4 Summary

The proposed Class of problems is multi-disciplinary, which combine both the structural and dynamic modelling techniques. Structural modelling tools are good candidates for modelling the deterministic part of a system, which as we stated are the functional and structural requirement styles. This provides the ability to separate the data with the process of treating the data. But in order to model the emergent behaviour, the structural model requires some aspects of probabilistic and statistical methods. Probabilistic techniques provide predictive information on the non-deterministic nature of distributed system and this can be achieved by simulation and prototyping mechanisms. Statistical modelling provides the mapping through traceability matrix to blend the probabilistic with the deterministic. We integrated the blended modelling approach into the architectural and design phase of the proposed research and development process model.

Figure 36 reflects the change in the engineering discipline of messaging infrastructures which is a consequence to the change in the cultural and technological modus operandi. The diagram is a high level model of the blended approach and shows how to use several modelling styles to define the different types of requirements. The inductive modelling styles are applicable to the behavioural and communication style of the requirements, which resolve to the non-deterministic character of the requirements. The deductive modelling techniques are employed for the structural models which forms part of the data and structural styles of the requirement.

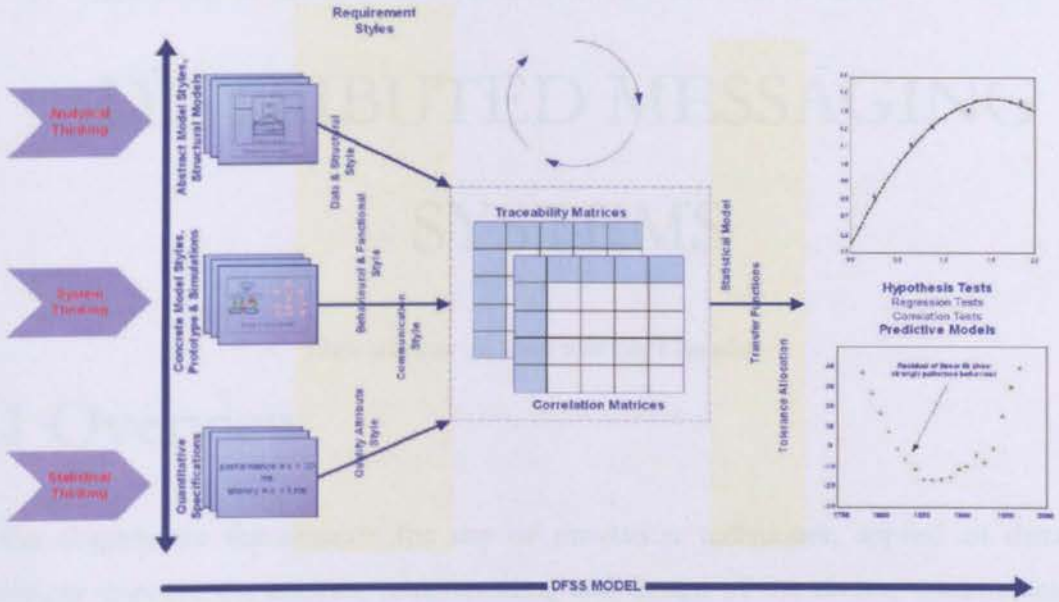

Figure 41 Blended Modelling Approach

Since the blended modelling approach comprised of several techniques we critically evaluated the different tools for modelling, simulation and implementation of distributed messaging systems. To add rigour to the evaluation process, we employed the AHP method to quantitatively assess the tools against the requirements and their conformance to the development of distributed messaging systems. The AHP enabled us to check the consistency across multiple decisions making criteria. We justified the choice of the tools by evaluating several inputs from expert groups, literature review of the state of art and industrial research / surveys to ensure that the consistency index and consistency ratio of each criterion is within 15 % and 10 % respectively. This is an indication to validate the integrity of the outcome from the evaluation analysis.

4 DESIGN & SIMULATION OF DISTRIBUTED MESSAGING SYSTEMS

Those who can do, those who can't simulate

4.1 Overview

In this chapter, we demonstrate the use of simulation techniques, applied to dynamic models, to improve the analysis, understanding and design of the diverse communication models within a distributed messaging system. To carry out the simulation experiments, we investigated an actual messaging system called the Virtual Mobile Redirector (VMR) (VMR02a and VMR02b) and analysed its communication dynamics, employing Coloured Petri Nets, as a simulation tool.

The rationale behind the choice of simulating the VMR application is due to the fact that it is a real messaging system, deployed within the Telecoms network but it has never have been properly assessed for quality, although traditional software testing has been carried out. The VMR provide an insight into the functionalities and operation of messaging gateway and despite the fact that it was designed using a Multi-Agent Architecture, the application was never deployed on multiple processors or virtual machines. This chapter is set in two folds, whereby the first set of simulation experiments analyse the communicating agents of the VMR on a single processing unit and the second series of investigations examine the complexity of **migrating** the VMR application from one virtual machine to a distributed architecture with several interconnected processors. In doing so, the management of distributed nodes, the communication model of each individual VMR agents, the concept of cluster management and fail over strategies have to be investigated which leads to the concept of **service discovery**.

The VMR agents communicate using interconnected queues which are typically the most commonly used data structures within the implementation of communication mediums

(substratum) for distributed systems. Queuing theory is the mathematical study of queues within a system which is essentially applied to analyse attributes such as performance, robustness and reliability of multi-programmed computer systems (JACK57, Gor67, Sche67 Buz71a, Buz71b, Buz73, Bask75).

In 1971, Moore showed that queuing network models could predict the response times on the Michigan Terminal System (MTS) to within 10% (Moor71). Since, rigorous validations have verified that these models reproduce observed performance with significant accuracy (Buze75, Giam76, Hugh73, Rose78, and Ram90). Many analysts have experienced puzzlement at the accuracy of queuing network results, thus the popularity of queuing networks grew to become the Markovian queuing network based on assumption of queuing theory (klei75 and Klei76).

Queuing theory is the result of extensive empirical research in the domain of waiting lines and provides us with theoretical models that explain the behaviour of queues within certain constraints (Moor71, Gross98). Nevertheless queuing theory is often mathematically restrictive to enable the representations of real-world phenomenon and complex systems (Nik03). This restriction arises because the underlying assumptions of the theory do not always hold in the real world; the mathematical models often assume infinite numbers of customers, or queue capacity, or no bounds on inter-arrival or service times, when it is quite apparent that these bounds must exist in reality. In other cases the theoretical solution may either prove intractable or insufficiently informative to be useful, hence alternative means of analysis have been devised in order to analyse problems which do not fall under the mathematical scope of queuing theory. Those means are simulations and dynamic modelling techniques which have become very popular to analyse highly collaborative software systems.

4.2 Simulation of the VMR Agents within a Distributed Messaging System

In the last three decades research in computer science strongly contributed to the generation of simulation tools and showed considerable interest in the development for dynamical

simulations of stochastic processes (Chan86, Lub87, Kha93, Kha96, Lub00, Lub01, Kor02 and Novo03). One of the primary motivations is the desire to carry out simulations over larger length and time scales, e.g. rigorous parallel Metropolis Monte Carlo (MC) simulations (Hai95). In recent years, several formalisms and related automation tools in software engineering have been developed to incorporate simulation capabilities. Within the wide spectrum of simulation techniques, we were required to focus on a selected subset, either for historical reasons or recently achieved popularity that best conform to the CTQs needed to simulate a distributed messaging system, which has been achieved by the AHP analysis in chapter 3.

As mentioned earlier, we devised a case study based on the functionality of the VMR. Essentially, the VMR enables mobile phones to connect to IP based application through protocol conversion which is designed on a multi-agent system. The application consists of different interconnected software agents, wherein an agent encapsulates a series of methods as one class function and comprises of input/output components expressed as "sensors" with the primary purpose to receiving and dispatching packets of information via queues. The agent also contains a processing unit, which identifies which methods are to be triggered when the sensors receive data.

The VMR represents a logical organisation of agents, which interact in a common environment to achieve a particular goal. It focuses on the collaborative resolution of global problem by a set of distributed entities. Like typical Multi-Agent System (MAS), the VMR abstracts the interaction between individual agents, defining the independency of the underlying architecture, which can be deployed on both centralised and highly distributed systems.

4.2.1 Description of the Virtual Mobile Redirector (VMR)

The VMR architecture has been established on a hierarchical structure of a Multi-Agent System (MAS) which can be decomposed into three levels of controls; 1) the agent level, 2) cluster level and 3) system level. Each level has different structures, activities and mechanisms, but the levels interact with each other during local / global communication.

4.2.1.1 Agent Level

The generic model of a VMR agent can be regarded as a class, which is made up of three sub classes; Head Automata, Sensor and Action class (*see Figure 42*). The Head Automata is a generalisation of three smaller classes. In this work, we focus on the Logic, the Agent Meta Knowledge and Sensor modules only, since the BDE (Belief Desire and Experiment) module encapsulates aspect of learning and artificial intelligence that has not been considered in the problem domain of this study.

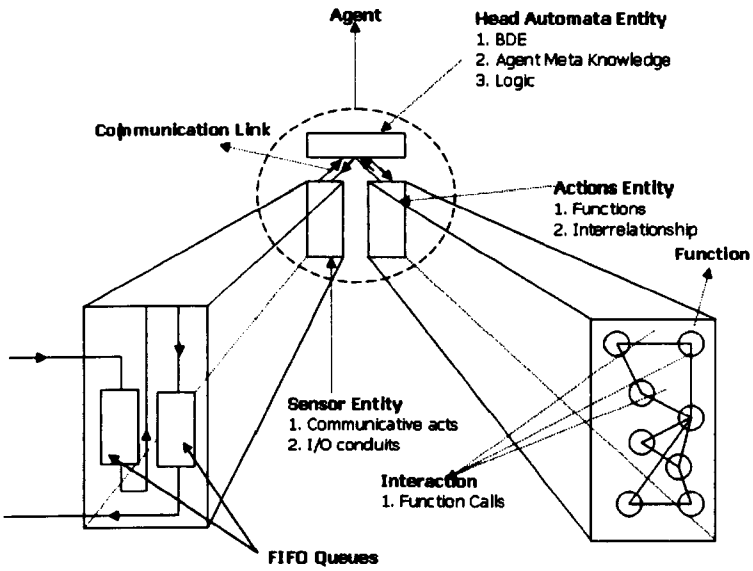

Figure 42 VMR Agent Model

The Agent Meta Knowledge class contains attributes that inform an agent of its basic properties and code of behaviour. The information is usually fed at boot time and consequently requires an exchange of data packets from an external source, which is a process that has to be included in the simulation analysis. However, for the purpose of simulating reliability within the system, we had to extend the perimeter of the Meta Knowledge class to include utilities that allow the agent to observe its own behaviour during failure and repair.

The Logic module encapsulates method that holds conditions in order to take the necessary steps when an incoming transmission reaches the agent. Its job is to be notified by the Sensor class module of a request, checks with the Meta Knowledge class, and hence decides which function to trigger from the Action class.

The Sensor class holds a series of communicative acts (protocols), which are embedded in different methods that establish whether to accept or reject an incoming packet as well as identifying priority rights of data packets. The I/O conduit component of the Sensor class points to two FIFO queues which reside at the input and output ports. Finally the Action class module is a normal class that holds methods that waits to be called by the Logic Class Module of the Head Automata.

4.2.1.2 Cluster Level

Moving a level up in the hierarchy, above the Agent level, we model the cluster board. VMR agents with identical or similar tasks are grouped into clusters to optimise communication among them, as shown in Figure 43.

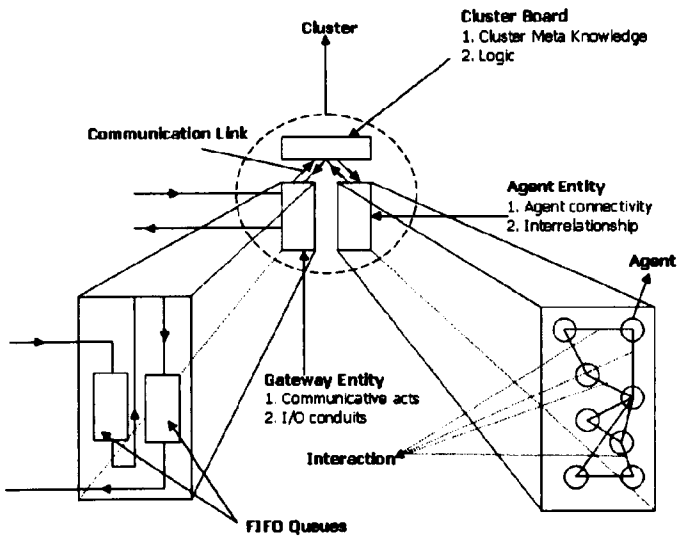

Figure 43 VMR Cluster Model

As it can be observed, the structure of the cluster is similar to the Agent architecture (*see Figure 42*). A VMR cluster is a class which is made up with three sub classes; the Cluster Board, the Gateway and group of Agents. The Cluster Board holds two classes, which are the Cluster Meta Knowledge and the Logic component. The Cluster Meta Knowledge module has information about the common objectives of a cluster that co-ordinates the agents' behaviour inside the cluster. Similarly to the Agent entity, we expanded the Cluster Meta Knowledge class to support a component that will study the mechanism of failure of a

cluster. These mechanisms relate to the aspect of Service Discovery upon which a cluster management system is to be deployed, as presented in chapter 4.

The Logic class of the cluster board carries out similar functions as those of the Logic class of the Agent Level, where the distinction is on the conditional methods. Hence when a message arrives via the Gateway class, the Logic class consults the Cluster Meta Knowledge class and decides which agent to be triggered from the agent pool. This is represented by the Agent class which holds a matrix mapping the routes and connections of different agents. Depending on the decision of the Logic module, the appropriate agents are called to action. The Gateway class acts in similar ways to that of a Sensor class of the VMR Agent. It holds two queues outlining the I/O conduit and the communicative acts that emphasise the cluster interaction protocols.

4.2.1.3 System Level

Finally, at the highest level of the VMR architecture, there is a system that represents the entire society (known universe) into which clusters of agents communicate under the supervision of the universal board.

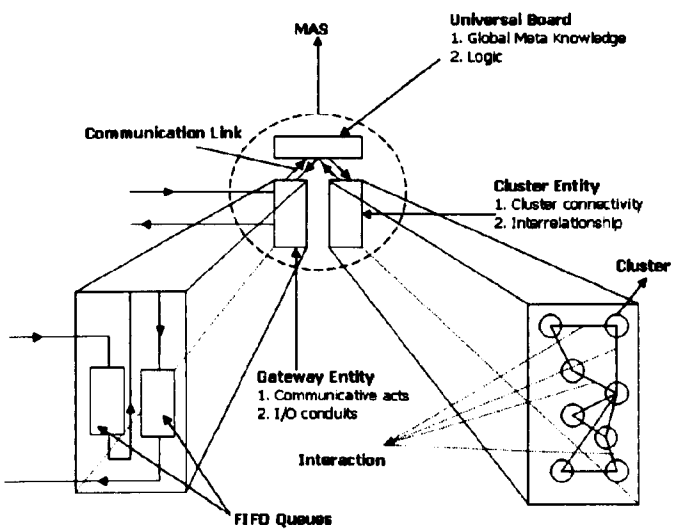

Figure 44 VMR MAS Model

In Figure 44, we represent the architecture of the entire MAS of the VMR. The MAS holds a group of clusters sharing common objectives together which consists of three classes; 1) Universal Board, 2) Cluster and 3) Gateway class. The description of the gateway class is

similar in terms of property and behaviour to the Gateway class of a cluster and thus omitted. However the Meta data, describing the MAS, differs from the Cluster and Agent level and the conditional methods inside the Logic class may also differ depending on the environment variables and rules of the MAS.

The fact that the VMR was designed in a hierarchical manner, it enabled us to design the simulation of the model based on the hierarchical features of Coloured Petri Nets, though the process is not a one to one mapping, it still helped in the design translation process. The presence of hierarchy is technically a state space reduction method that exploits the common patterns found in distributed and concurrent designs at the different levels of hierarchy. The basic idea is to factor out the common patterns; hereby obtaining a reduced state space that is typically orders of magnitudes smaller that corresponding full state space. Until now, the construction of the reduced state spaces has required the user to implement in ML an equivalence predicate to determine whether two states are similar or not. Experience has shown that this is a complex, error prone and time consuming task. To overcome this problem, improved tool support has been developed that allows the user to write high level specifications by annotating the colour sets of the CPN model. The equivalence predicate is then automatically generated from this high-level specification.

4.2.2 CPN Simulation of the VMR Application

The communication model of the VMR agents are supervised by the communicative acts of the sensors, acting as interfaces. The sensors are made up of two FIFO queues, into which there are several critical parameters. These parameters are analysed and ultimately managed using a technique called Critical Parameter Management (part of DFSS) (Slee06 and Crev03). In order to further analyse the VMR application we build a parameter diagram (*see Figure 45*) to classify the key factors that are entering and exiting the system.

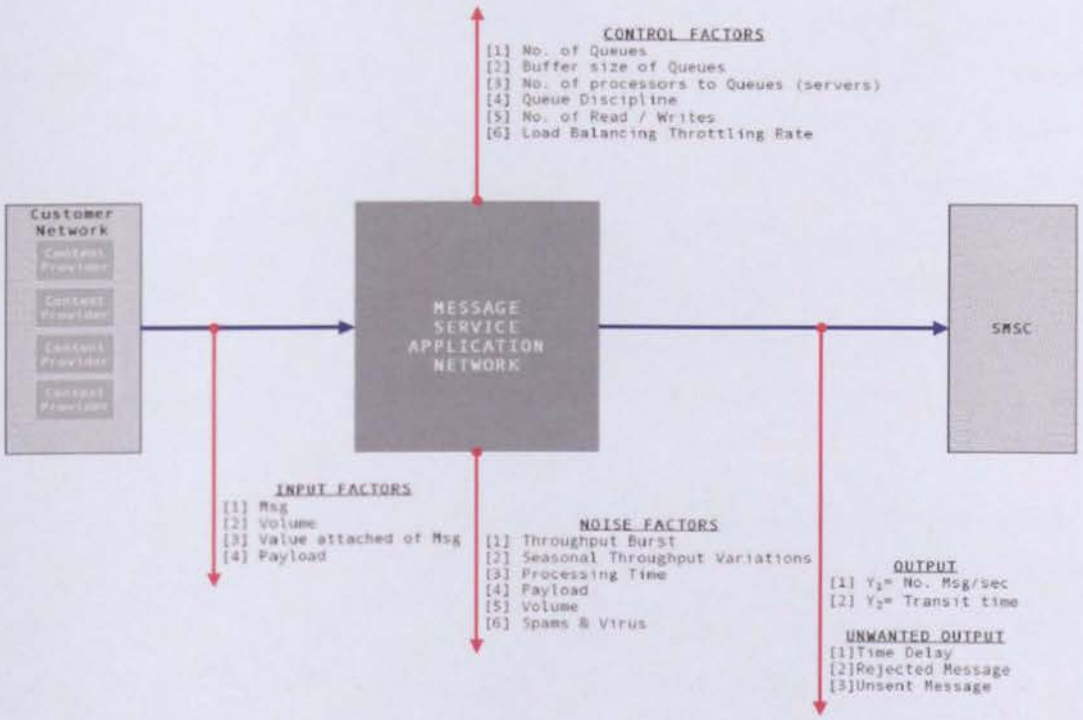

Figure 45 CPM in Queuing Based Network

The critical parameters are categorised into classes: 1) control factors (*parameters that can be controlled and measured*); 2) input factors (*parameters that are measurable but not controllable as they are usually from external systems*); 3) noise factors (*parameters that are not measurable and most of the time are unknown*); 4) unwanted factors (*parameters which are against the CTQs of the customers and need to be eradicated*) and finally 5) output factors (*parameters that exit the systems to be delivered to the customers*). In our experiment we focus on the parameters which are essential to develop a waiting line model for the sensor operation of each agent such as:

- The arrival distribution for the packets (Input factor)
- The service time distribution for packet processing (Control Factor)
- The queue discipline (Control Factor)
- No efficient management of resources, memory usage for the queue's buffer size (Control Factor)
- A burst in packet throughput result to an increase in packet loss and congestion at the edge of the queue (Output Factor)

These parameters form the interconnectivity model across the distributed systems which contribute extensively to the QoS and general health of the system and are characterised by a number of quality attributes. Firstly, system performance is a generic quality attribute which is achievable, in the Telecoms messaging context, by processing more messages requiring the system to reduce its internal latency and congestion rate. This will consequently result to an increase in the rate of incoming packets at the queues' edge. In the VMR situation, the latency can be measured using either the average time taken for a given message to be delivered to a processing server via a queue or the average amount of time a given message waits in the buffer of a queue.

Secondly, the system reliability is defined by the reduction of packet loss and out-of -time packets. Packet loss can be reduced by adjusting the rules governing the queuing behaviour of an agent, and out-of-time packets can be reduced by adjusting the number of processing servers to the queues, i.e. to decrease the processing time of a queue.

Lastly, resource efficiency also plays a major role to the economical viability of the system, attainable through intelligent management and dynamic sizing of queue's buffer size. Although this quality attribute is addressed manually in this chapter (through interactive simulation), the automated approach of resource management is discussed in details in chapter 5.

These three generic quality attributes become the objectives of the experiment with a goal statement which is formulated as follows:

"*To build and calibrate the queuing system of the VMR MAS to accepts maximum throughput at the minimum latency possible with the smallest buffer size possible (resource management) and minimum packet loss (reliability)*" Based on this hypothesis, we designed the simulation experiment of the VMR agents.

4.2.2.1 Functional Model

The list of functional requirements has been devised to characterise the functionality of each queue within an agent of the VMR system as shown in Table 3.

	Requirement ID	Functional Requirement Description
00	Req00	Each queue of the VMR MAS shall have a buffer size which can be dynamically configured.
02	Req01	Each queue of the Sensor class of any given VMR agent, cluster or system shall be ordered in a FIFO structure
03	Req02	The queues of the Sensor Class of a VMR agent/ cluster or system shall accept messages that are tolerable by the rules specified by the communicative acts
04	Req03	Each incoming message shall be time stamped with a ticket ID
05	Req04	Each message shall join the buffer order of the queue of the Sensor class
06	Req05	Each message shall wait in queue until claimed by a processing server of an agent

Table 3 Requirement Model of Queues within MAS

Non-functional requirements relate to the quality attributes or CTQs that are expected to be achieved from the functional requirements. In this study, we employ the FURPS model (an acronym for Functionality, Usability, Reliability, Performance and Supportability) (Chu03), developed by Hewlett-Packard Research & Development (R&D) labs in 1984, to model the CTQs of the VMR system. The FURPS is a technique that enables software designers to drill down the CTQ chart through specialisation of generic quality attributes such as performance and reliability and this is shown in Figure 46.

The FURPS model is a generic model for quality analysis which means that not all quality attribute it represents may apply to a particular problem domain. In our situation, we only looked at reliability and performance of the VMR MAS, since they are the common factors that are usually discussed in networking and communication models such as messaging infrastructures. The objective of the FURPS is to drill down from a generic quality attributes, such as performance or reliability, until measurable and controllable quality attributes are identified. In the case of performance, they are: 1) no. of packets per second for throughput, 2) transit time for latency and 3) processing time for latency.

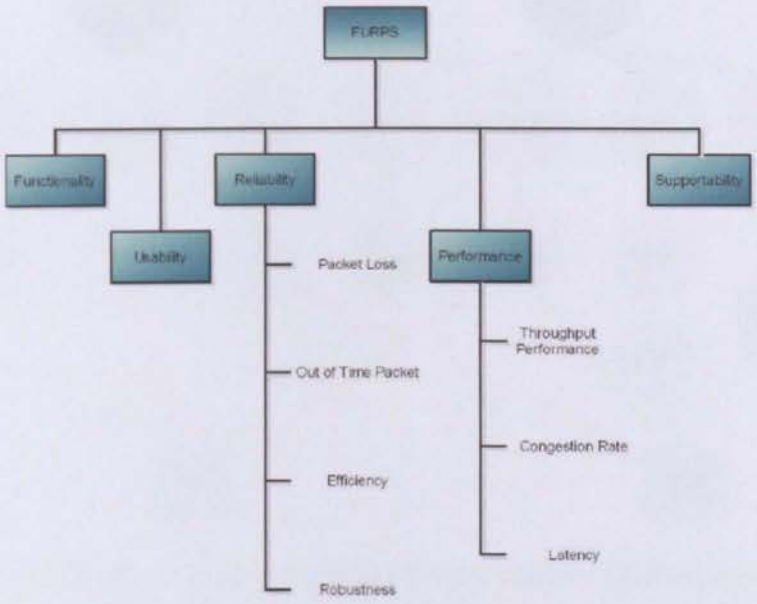

Figure 46 FURPS of the Queuing Models in MAS

4.2.2.2 Mapping CTQs to Functional Requirement

We employed the controllable CTQs derived from the FURPS techniques to analyse the relationships between the CTQs and the functional requirements and identifying the conflicting quality attributes, we model a CTQ Function Map as Figure 47 illustrates.

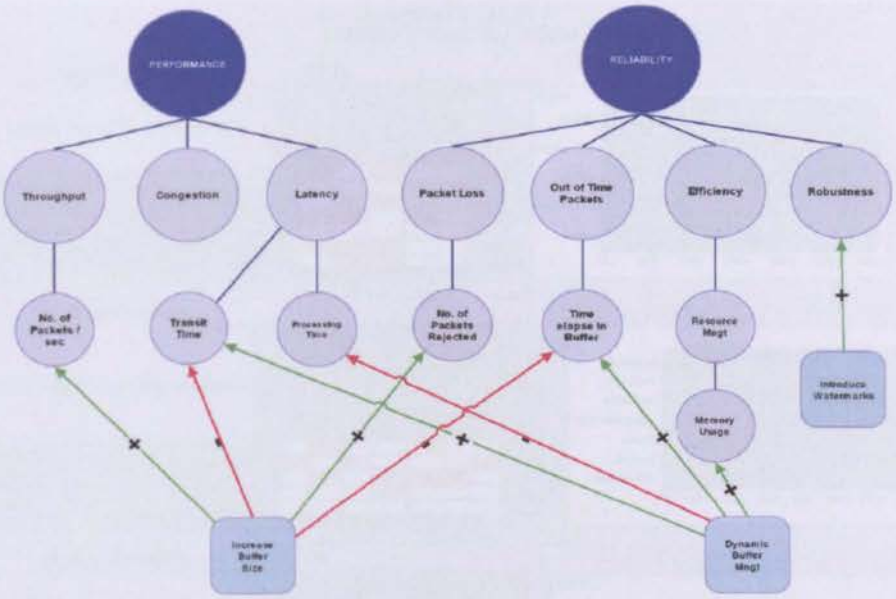

Figure 47 CTQs vs. Functional Requirement within a Queue based MAS

In order to ensure the highest performance and reliability of the queuing system, we implement dedicated functions, e.g. the function "*Increase Buffer Size*" has a positive influence over the CTQ, no. of packets / sec, which is marked with a "plus" sign in Figure 47. However, the same function negatively affects the reliability of the queue, in terms of out-of-time packet, because increasing the buffer size will result in increasing the elapse time a packet spends in the buffer of the queue (this is marked with a "minus" sign in Figure 47). The model provides a clear definition of the quality attributes, addressed by the system's functions and illustrates how they contradict or complement each other.

4.2.2.3 The House of Quality Analysis

The House of Quality (*see chapter 3*), is a robust tool that provides the matrix of mapping the CTQs with the functional requirements and allows a row by column CTQ drill down, as shown in Figure 48, which includes 3 levels of CTQ drill down from the high level quality attribute to the measurable ones.

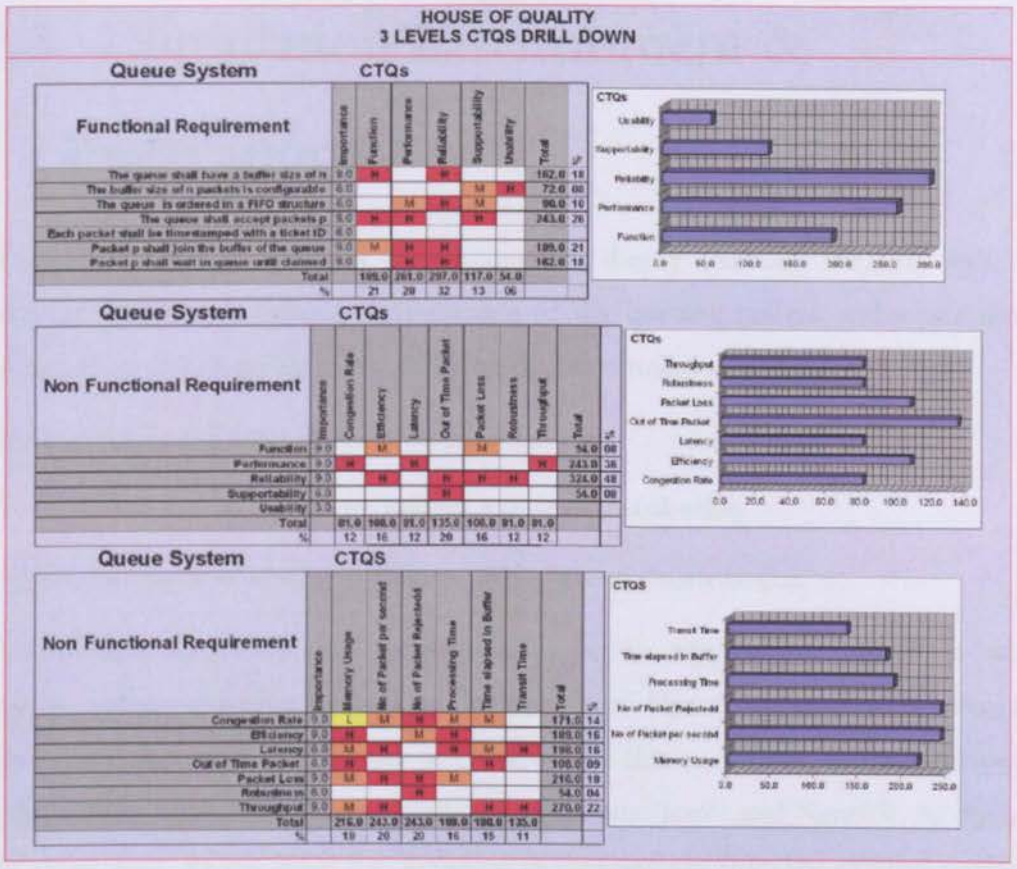

Figure 48 HoQ analysis of Queues within a MAS

As a result of exercising the HoQ, we were able to identify the most important CTQs which are also shown in the horizontal bar charts of Figure 48. In doing so, the metrics for the experiments have been revealed and comprised of the following:

- Arrival Rate is measured as number of packets per second
- Packet Loss is measured as the number of packets rejected over time,
- The transit time of a packet within a queue is measured by (delivery rate - arrival rate)
- Out-of-time packet is measured by buffer population above a time threshold

The objective of the experiment is to simulate Petri Nets of the VMR to gather knowledge and data on the aforementioned metrics which allows design decisions to be made on the models and directives formulated prior to implementation.

4.2.3 Simulation Environment & Observations

Based on the empirical research of the queuing theory (Ada02), we may now assign probabilistic distribution behind each variable of the queuing system and inject simulated results into these distributions to the CPN model for simulation purposes:

- The arrival rate follows a Poisson distribution
- The buffering rate of the queue follows a normal distribution
- The processing time of servers follows an exponential distribution

The distribution algorithms are parameterised, hence configurable, which is achieved through the parameterisation process of Coloured Petri Nets (Chr97). When making use of the hierarchical decomposition feature of CPN, we are able to describe the architecture of an agent-distributed system at different level of perceptions (Jen91 and Nem03). At the highest level of expression, we modelled the incoming of data packets from the external networks to the VMR system which is illustrated in Figure 49.

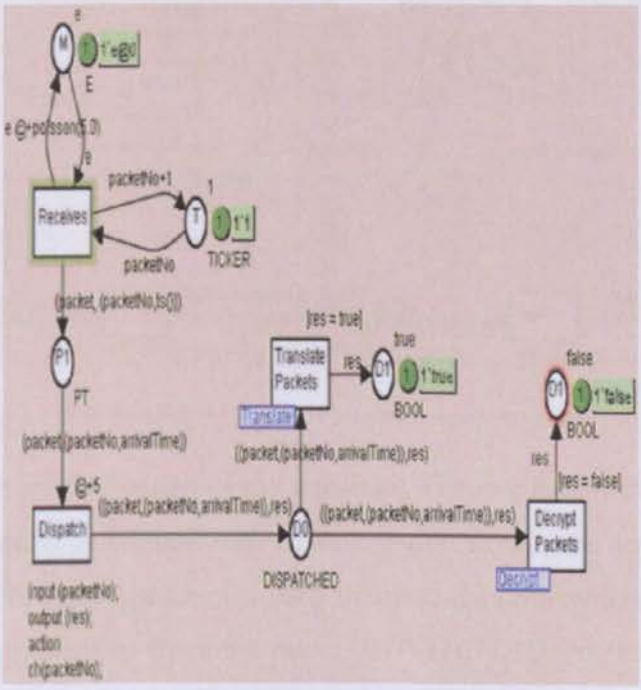

Figure 49 CPN Model of VMR MAS

The CPN model starts with the transition *Dispatch*, receiving one packet at a random time, which follows a Poisson distribution with a mean of *5.0 ms*, from an external source. Since the model is a timed Petri Net, variable *E* at place *M* is timed in millisecond whilst each packet is numbered at the place *TICKER*, thus the place *Dispatch*, knows about the packet number (*packetNo*). Next, at transition *Dispatch*, a function[2] *ch()* is triggered to randomly select between 2 transition operations, either *Translate Packet* or *Decrypt Packet* . These functions add more knowledge to the model; hence during simulations more complex processes can be investigated in an attempt to emulate reality as closely as possible. Upon exit of the transition *Dispatch*, the place *DISPATCHED* is fired. Inside the transitions *Translate Packets* and *Decrypt Packets*, there are a series of distinct software agents communicating to each other with a common objective. In the VMR system, operations like "*translate the packet structure to a general format*" are performed by a cluster of agents. In an attempt to simulate these functions, we constructed a hierarchical transition *Translate Packet* within which are a group of agents connected to each other, and the same design applies to the transition *Decrypt Packet.*

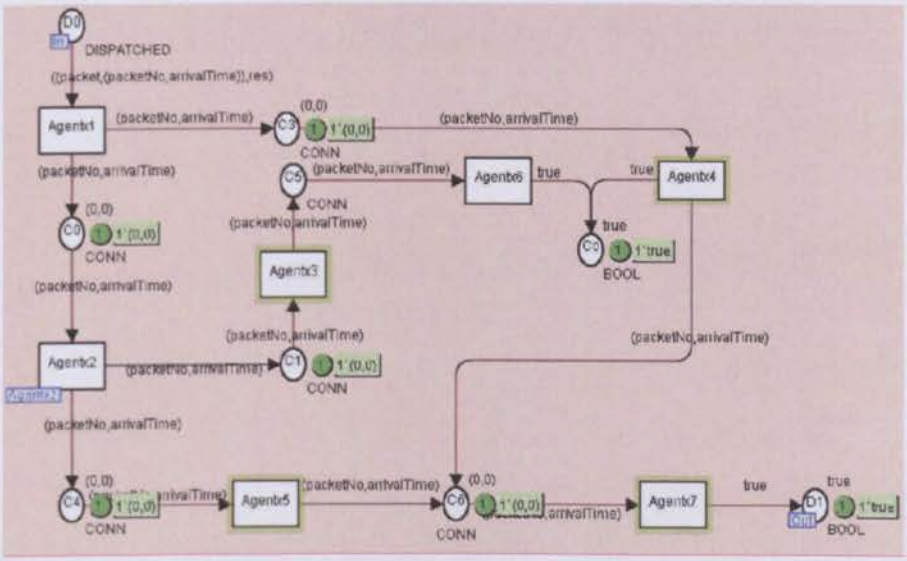

Figure 50 CPN Model of Agent Communication within the VMR

Figure 50, depicts the internal design of the transition *Translate Packet*, wherein, several agents are non-linearly assembled. Packets move concurrently across the agent system which is supported by the CPN infrastructure for such dynamic data movements. An agent receives the *packetNo* and the *arrivalTime* from the place *DISPATCHED*. As the packet travels from

[2] Incorporating functions to transition is feasible, by creating codes written in CPN ML

one agent to another, the task of translating a message to a generic format is distributed among various agents.

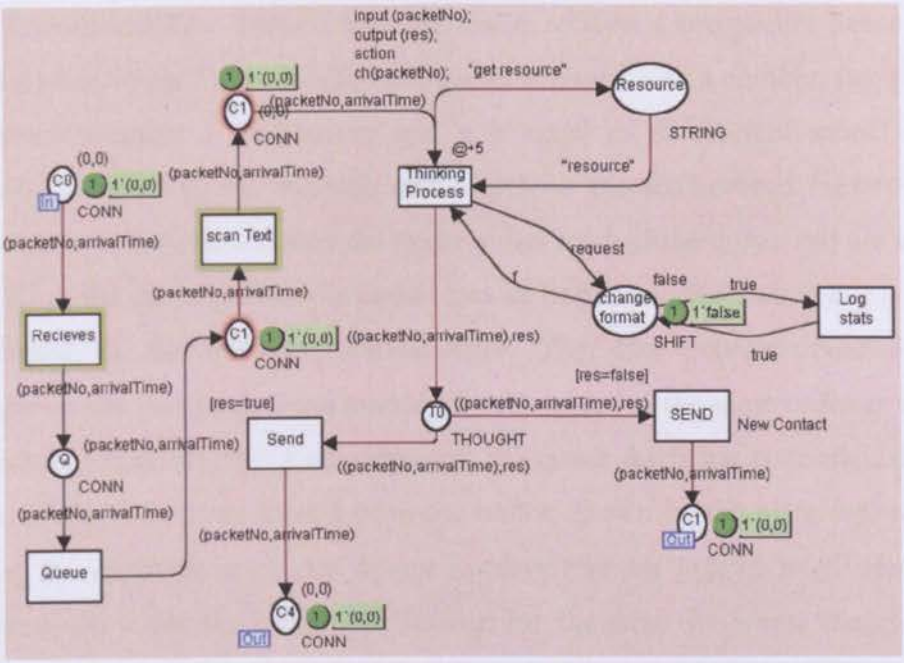

Figure 51 CPN Model of Agent Operations within the VMR

Drilling further through the Petri Net hierarchy, we find the design of one agent, which is illustrated in Figure 51.

The agent triggers several processes towards translating a packet into different formats. We model a transition called *Thinking Process* which decides the format a packet is to be converted into, which depends on the message types and properties, captured whilst scanning through the protocol structure.

At the lowest level of the hierarchy, we model the FIFO queue structure (*see Figure 52*), which is the communication medium of each agent, found in the Sensor class. In an agent we expressed the transition *Queue* which buffers all the packets reaching the agent. The queue buffer system has a value *vh*, being dynamic and adjustable during simulation, indicating the top stop mark capacity of the queue's buffer and when hit, it blocks the packet entry. Unless the buffer is freed, outgoing packets have to wait for their acceptance. Another watermark is the high watermark denoted by *h*, and when reached, the rate of incoming packet is decreased until the number of packets inside the queue's buffer reaches the low

water mark which indicates that the queue is within normal parameters of operation and no protective measures are required.

When the Transition *Packet Arrives* is fired, the queue receives a new packet, hence a token is added to the place *arrived*. Upon arrival, each packet is paired with a number, (see place *packet stamped*), which contains a time stamp and it is equal to the current model time. The timestamp is generated by the function *timeStamp()*, on the arc between *Packet Arrives* and *arrived*. The place *arrived* characterises the packets that reached the queue and are waiting for clearance. Next, the queue pushes the packet into its buffer and the waiting line is shown by the occurrence of the transition *PushIntoBuffer*. The arcs between *Next Packet* and *PushIntoBuffer* ensure that packets are loaded into the queue in the same order in which they joined which enforces the FIFO structure and the queue discipline is configurable during simulation. When packets are loaded onto the buffer, shown by the place *buffered*, they are counted (*see the inscriptions on the arcs between PushIntoBuffer and buffered, noOfPackets*). At this point, packets are inside the buffer and waiting for the next run where the queue might perform some simple processing to the packets before releasing them. Once a packet completes its processing cycle, a token is added to the place *leaves*. Any claims from the place, causes the transition *Claims Packets* to be fired which finally state that a packet has left the queue.

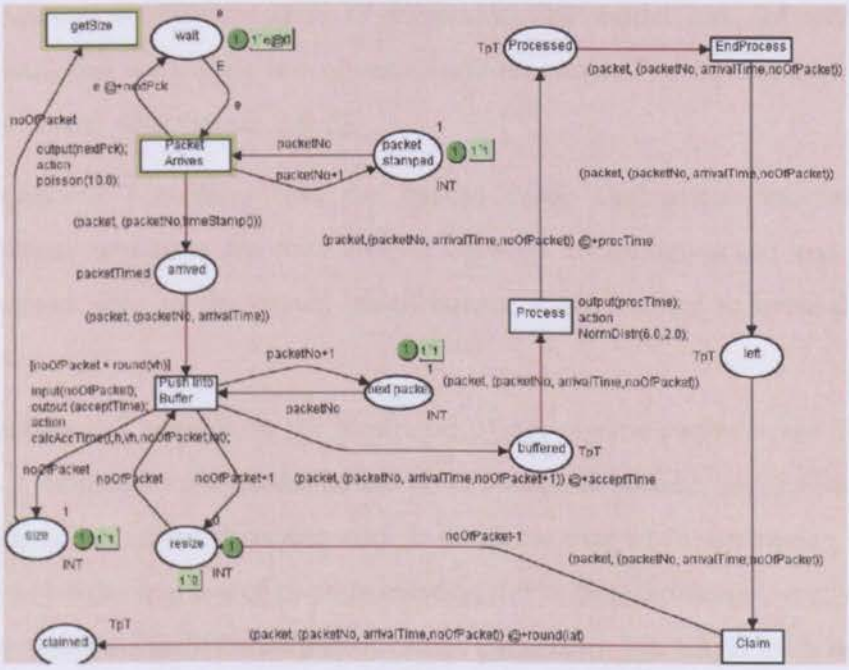

Figure 52 CPN Model of a Queue system within an Agent of the VMR

A Poisson function, at the entry point is used to model the inter-arrival time for packets, and it can be found in the code segment associated with *Packet Arrives*. By changing the value in the function's argument, the mean of the Poisson is altered and thus providing different spectrum of simulation conditions. The function *CalcAccTime()* in the code segment for the transition *PushIntoBuffer* is to calculate how much time is needed for loading a given packet. The function *NormDistr()* is used to calculate how much time is used for a packet to be processed, as shown in the code segment for *Process*. This is particularly useful when one needs to make a number of simulations using different parameters.

In summary the objectives of the simulation of the VMR CPN model are as follows:

- To observe the percentage time a unit (packet) leaves an agent given a high input rate. This is to probe the information on the throughput of the queue during high input to the VMR system.
- To observe the average time a unit (packet) spends in the VMR System. This is to provide information on the likelihood of out-of-time packet(s), should the latter wait excessively in any queue buffer of any agent.

Observations

The simulation exercises investigate how packets move amongst the VMR Agents, with variable parameters at different level of hierarchy. The model was fed with packets at random intervals and the output was observed and recorded. The experiments were broken down into two series (experiment 1 & 2).

- **Experiment 1** explains that the queues inside the agents resorted to protect themselves whenever the time interval between incoming packets was smaller than the service time of the queue, which consequently resulted in some packets being rejected.
- **Experiment 2** stresses on the likelihood of out-of-time packet in the VMR System. This is defined by a function of the agent's internal latency, queue's buffer size and the input rate at the receiving end. It confirms that while attempting to gracefully slow down the number of packets entering the buffer, should an overflow occur; the queue inside an agent consequently causes packets to wait longer in its buffer.

Experiment 1 starts by investigating the percentage unit time (*ms*) a packet leaves the multi-agent system given a burst of input. The objective is to observe the throughput of the system in unconditional circumstances and evaluate its behaviour and efficiency to protect itself. We run through three simulations of *4000 steps* and at step *1000* , we changed the *Poisson mean* from *1 packet per 10ms* to *1 packet per 3ms*, which provided the scenario for the burst and is the type of situation common to the domain of messaging. At each simulation we decreased the value of the watermarks, thus decreasing the waiting line in view of achieving conditions where packets have to wait longer to be serviced.

Figure 53 runs through the simulation as the buffer sizes of the queues inside the agents were increased. We sampled the data into batches of *100 ms* interval and recorded the number of packets entering and leaving the queue. Given a burst in packet entry, the graph shows the difference between the input rate and the output rate. The close proximity between the input and output rate justifies that that the service rate of queues within the VMR system is adequately lower than the input rate given an input rate of *100 packets per second (1 pck/10ms)* and a burst of approximately *350 packets per second (1 pck/3ms)*. The simulation validates both decisions on the buffer size of lower limit *10* and upper limit *20*.

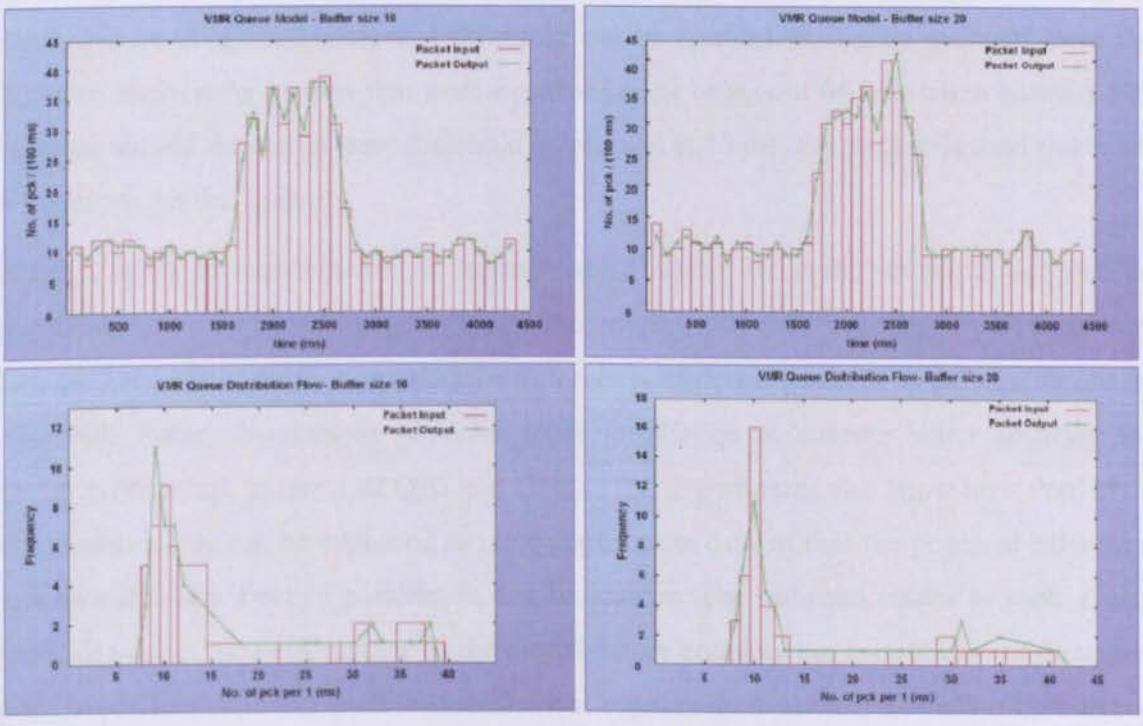

Figure 53 Observation of queues performance within the VMR MAS

Experiment 2 investigates the period of time a packet stays in the VMR at a given queue of an agent and draws the distribution of the packet's lifetime as the histogram in Figure 54 depicts.

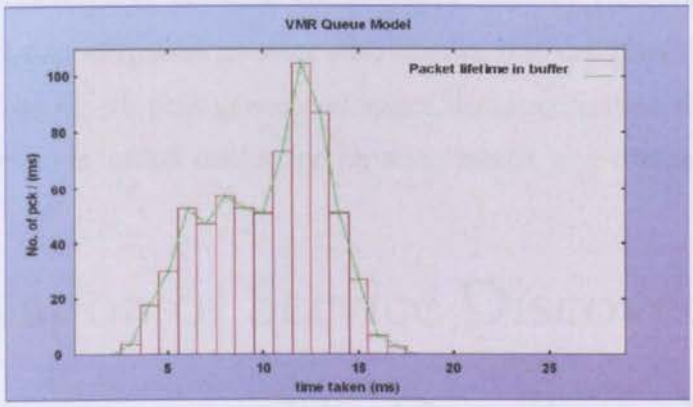

Figure 54 Queue Buffer Population

The graph reports on the time taken for a large sample of packets to leave any queue of an agent. The objective was to estimate the number of out-of-time packet that exists given an input burst and a decrease in the buffer size. The graph shows the period of time a packet takes to leave an agent, e.g. over hundred packets take *12 ms* to leave the queue of a VMR agent. Hence using such analysis, a threshold can be established, usually gathered from the VOC, to identify the packets that have a probability of being out-of-time when claimed. For instance, should the out-of-time threshold be marked at 15 ms, any packet beyond this mark is irrelevant for the system.

Given a series of requirements, at the early stage of the life cycle, we found out that by translating these requirements into Critical Parameters, and then injecting these parameters into the Petri Net simulation models, the different behavioural patterns of the models can be observed. These observations provided more knowledge to enforce better accuracy for quality estimations, in terms of QoS and CTQs. The experiments also show how Petri Nets as a methodology can be exploited to mine appropriate data so that the potential behaviour of a model under a set of parameters can be known. The obtained results provide a very good idea about the performance of the model, hence enabling the formulation of directives prior to product deployment. If we look back at experiment 1 where a discussion on burst of input rate is presented, this is often very true in the Telecoms and messaging systems. For instance, if we consider the message entry to a SMS Centre (SMSC) for one day, there is a

pattern, wherein bursts can be found during the morning when people wake up and starts sending text message and burst at lunch time during their break time. To deal with the surge, system designers over-estimate the buffer size of queues within their systems, which is very inefficient in terms of resource allocation and economic viability. However, using simulation based on historical data of typical message distribution, one can provide a range, a lower limit specification size for off peak period and upper limit specification size for peak period. This is type of design is called design for capacity, which is a competitive advantage if possessed.

4.3 Simulation of Service Discovery Mechanisms within Distributed Messaging Systems

In this section, we examined the work involved in deploying messaging architectures, such as the VMR, over a distributed system. We focused on tasks that are mainly linked to the aspect of managing the distributed systems, in term of connection management, inter-communication models, clusters management, resource distribution and enabling fail over strategies. To achieve these processes, we are required to model the features of service discovery within the distributed architecture.

The word discovery, as the oxford dictionary illustrates, is the act or process of finding out or becoming aware of what was yet not found (Oxf03). In the context of the network engineering, service discovery is a method of determining and instantiating the resources required to manage and operate distributed software entities. These software entities are often deployed across multiple nodes within a cluster wherein they communicate with each other to mutually form a software application and, in our case, to ultimately build up the message gateway application.

Existing models to service discovery have been developed primarily for fixed network backbone environments and typically rely on centralised components being accessible to potential service clients at any given time (Gutt99, Arn99 and Mic99). In highly dynamic

nature of underlying network topology, these models lack the designated service infrastructures, hence rendering such discovery mechanisms unsuitable for mobile ad hoc environments.

The concept of service discovery is well established in distributed systems, since networked entities need to discover available remote resources (Mull85). Work on service discovery in mobile ad hoc networks focus on using decentralised architectures to overcome the limitations of traditional discovery mechanisms, such as Service Location Protocol (SLP) (Gutt99), Jini (Arn99) and UPnP (Mic99), which rely on fixed infrastructure. Research in service discovery architectures for mobile environments can be classified into lower layer service discovery protocols (Haa99, li00 Xue01 and Koz03) and higher-layer service platforms (Herm00, Chak02 and Hel02). Service discovery protocols emphasise on efficiency and distributed infrastructure while service platforms focus on providing a middleware layer that enables applications to use a service oriented programming model. The reader should note that the works carried out on service discovery architectures, (Zhu02, Cho05 and Enge05), provide a survey and an evaluation analysis of different service discovery models for mobile ad hoc networks.

In our study we place emphasis on the higher layer service platform, with the objective of designing and simulating the different service discovery strategies in the context of distributed messaging systems. The aim is to model and build **fast** and **robust** service discovery strategies, which are the main drivers for efficient management and organisation of distributed system. Similar to VMR analysis, here we are also confronted with conflicting quality attributes i.e. **fast** (***performance***) vs. **robust** (***reliability***), which is addressed using quality modelling techniques.

4.3.1 Structure & Data Model

The data model explains the relationships between the known entities, showing how services are provided by a given application (*see Figure 55*). It also shows how the participants or software components and the physical nodes relate to each other within a cluster by means of service discovery strategies.

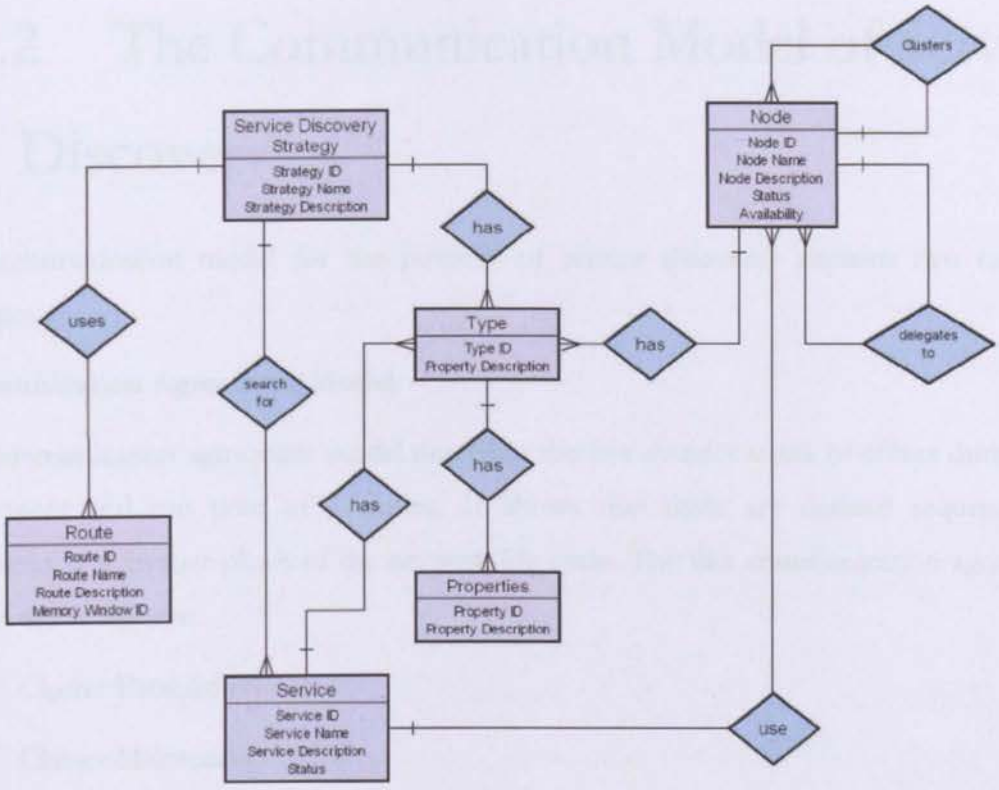

Figure 55 ERD of the Service Discovery Model

The Entity Relationship Diagram (ERD) in Figure 55, shows that the entity Service Discovery Strategy has many types, (entity Types) which defines the properties of a distinct strategy and the cluster of nodes they manage. The purpose of the Service Discovery Strategy is to locate a particular service, represented by the association *search for* entity Service. This association will define the route, shown by entity Route, which will be used by a node to process a particular service. The entity Route describes the connectivity model between nodes and services and its attribute *MemoryWindowID*, relates to the memory space allocated for a particular service which is managed on a distributed shared memory model. This is further elaborated in the communication model presented in chapter 6.

4.3.2 The Communication Model of Service Discovery

The communication model for the purpose of service discovery explains two essential concepts.

Communication Agreement Model

The communication agreement model describes the five distinct states of affairs during the deployment and run time of a cluster. It shows that there are defined sequences of operations at a distinct phase of the network life cycle. The five communication agreement models are as follows:

- Cluster Establishment (CE)
- Cluster Maintenance (CM)
- Service Agent Establishment (SAEST)
- Service Agent Enablement (SAEN)
- Service Agent Maintenance (SAM)

Communication Styles

In chapter 2, we introduced the concept of communication styles and explained how they influence the communication medium (substratum). In this section we show how the communication style defines the manner in which information is exchanged and how it determines the service discovery strategy. We considered the four most common strategies of service discovery:

- Exhaustive search
- Broadcast All acknowledge
- Broadcast 1 acknowledge
- Transactional publish subscribe

For each communication agreement model, a given service discovery strategy is tested against a set of quality attributes or CTQs which are the measurable using simulation and validation against the functional requirements. The CTQs are formulated in the requirement model phase, underlining their relationships with the functional requirements by making use of the HoQ matrix. The number of simulations required to complete the matrix is expressed by the equation $S = q(CS \times CAM)$, where S is the total number of simulations, CS is the total number of communication styles or service discovery strategies, CAM is the total number of communication agreement models (cluster states) and q is the total number of CTQs used at each cell.

The hypothesis states that the implementation of a particular communication style depends on the type of communication agreement model in place which implies that using only one type of service discovery strategy e.g. broadcast, to manage all types of communication agreement models may be not efficient and eventually implausible. The study has been designed to validate the hypothesis and applies a blended modelling approach to join simulation models with the qualitative modelling techniques.

4.3.3 Communication Agreement Model

The communication agreement model represents the distinct states of a cluster within a defined life cycle. It holds a collection of operations which are enforced by a cluster manager in order to start, run and maintain a system, in our case a messaging gateway. We have identified 5 distinct collections of operations and each of them is carried out using one or more service discovery strategies. Subject to the type of operations in place and the state of a cluster, each of the service discovery strategies has different properties with regards to the overall quality of a system.

Cluster Establishment

In order for a cluster to be established, the service manager initiates the start-up sequence within a physical node, and attempts to discover the availability of other nodes within the cluster (*see Figure 56*). It establishes a communication channel between each active node and these allow for the transfer of *heartbeats* between Service Managers as well as updates regarding service agent establishment and service agent maintenance messages across the

cluster. If more than one service manager is active, a mechanism should determine a Master Service Manager for the Cluster and information from the master should be distributed to Slave Service Managers. Both master and slave service managers are Local Service Managers for their respective nodes. Should no other service managers be found during the start-up sequence, the started service manager begins service agent establishment.

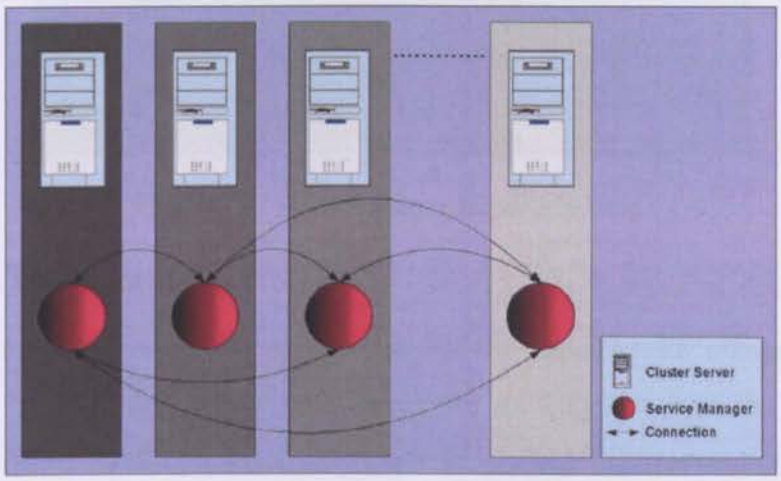

Figure 56 Cluster Establishment

Cluster Maintenance

In order for a Cluster to be maintained, the master service manager should be capable of handling the introduction and removal of slave service managers (essentially the addition or removal of nodes from the cluster). Should the master service manager fail, one of the remaining (if any) slave service managers should adopt the role of a master service manager. This may result in service agent establishment, enablement and maintenance as necessary which are described below.

Service Agent Establishment

In order for a Cluster to be fully functional, the Local Service Manager should be capable of starting Service Agents upon the node as determined by the Master Service Manager (*see* Figure 57). The Local Service manager controls different types of service agents which are responsible for distinct tasks. For instance a particular Service agent may be responsible for managing the connection to external entities such as SMSCs, whereas others are responsible for load balancing or routing.

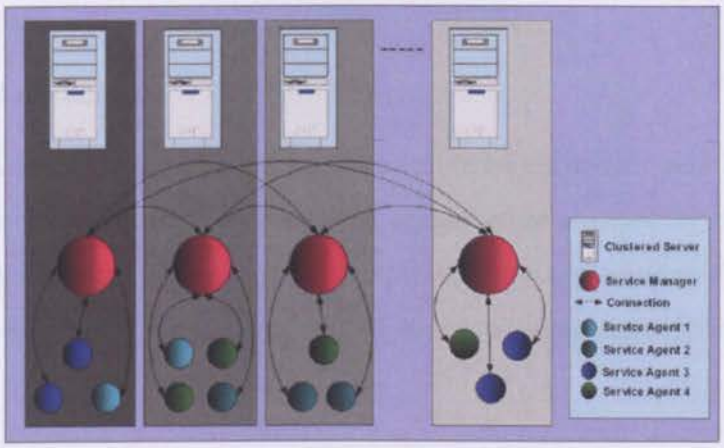

Figure 57 Service Agent Establishment

Service Agent Enablement

Service agent enablement is the process wherein the local service managers, instantiate service agents on individual nodes and report the start-up status to the master service manager. The master service manager will then issue routing information to the local service managers, which is then propagated to the service agents on each node. The master service Manager upon validation that the minimum required agent base is available to run an application instance i.e. a message gateway, should issue a signal to the local service managers to inform the service agents to (1) accept application connections; (2) establish SMSC/MMSC connections and (3) begin message processing. Figure 58, shows an example of service agent communication across a cluster and the communication channels between service agents. The communication between the service managers' communication and service agents is not shown for diagram simplicity.

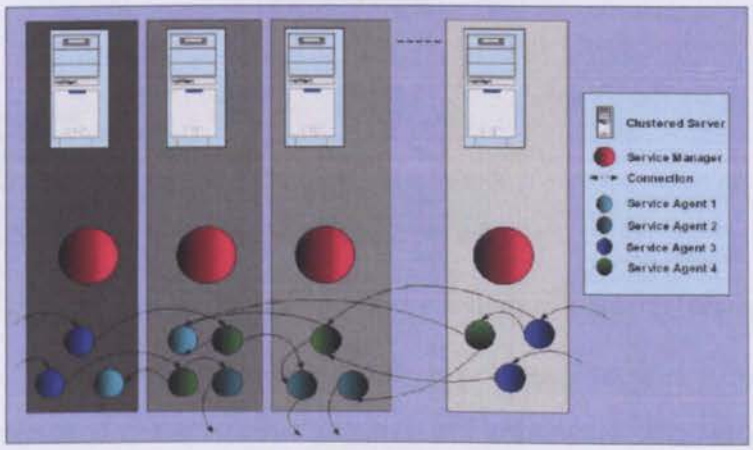

Figure 58 Service Agent Enablement

Service Agent Maintenance

Service agent maintenance is performed by a local service manager and is responsible for handling the start-up and shutdown of service agents as determined by the master service manager. The Local Service Manager is also responsible for reporting to the Master Service manager any failure of a Service Agent so that it can be instantiated elsewhere within the cluster.

4.3.4 Communication Styles

We define communication styles as the ways or manner information is conveyed across dispersed nodes within a cluster. As regards to service discovery, communication styles are the different strategies applied to locate and instantiate software entities within a cluster. As mentioned earlier, the discovery strategies are available with various properties and can be broken down into the following categories:

Exhaustive Search Strategy

Exhaustive search is a brute force method where a service polls for information against the known services until the desired response is received (*see Figure 59*).

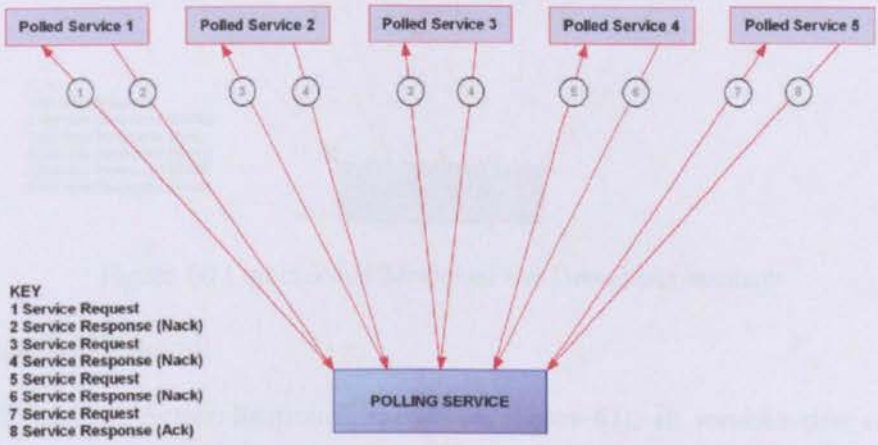

Figure 59 Operational Model of the Exhaustive Search Strategy

With an exhaustive search any service will iteratively poll other services starting with the first, wait for a response and if the result is negative will continue to the second and await its

response until such time as the desired response is obtained or no further services to be polled remain.

Broadcast Strategy

Broadcast discovery is a method where a polling service broadcasts a request to a number of services in order to receive a response. This is normally an information request or a request for something to be processed. There are two models which can be considered:

Broadcast (All Respond)

With the "Broadcast All Respond" model, all services that can possibly handle a request are transmitted and each polled service issues a request response (*see Figure 60*). This request response can be either a positive response indicating a service has been performed successfully or an information response passing data to the polling service. It can also be a negative response indicating that the service request cannot be performed or the information requested is not available or accessible by the polled service. The polling service should have a timeout mechanism if only unsuccessful responses or no response be forthcoming.

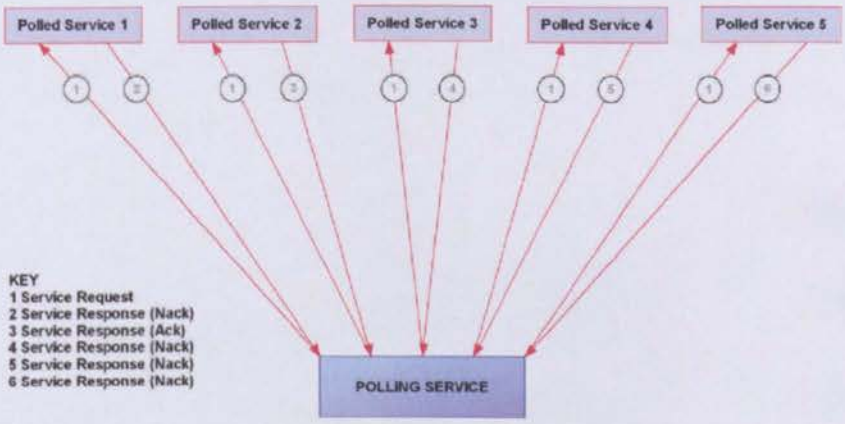

Figure 60 Operational Model of the Broadcast Strategy

Broadcast (Active Respond)

With the "Broadcast Active Respond" model (*see Figure 61*), all services that can possibly handle a service request are transmitted however only polled services that can issue a positive response indicating a service has been performed successfully or return data to the polling service, responds. All other services continue as if the service request was never received. The polling service should have a timeout mechanism if no response be forthcoming.

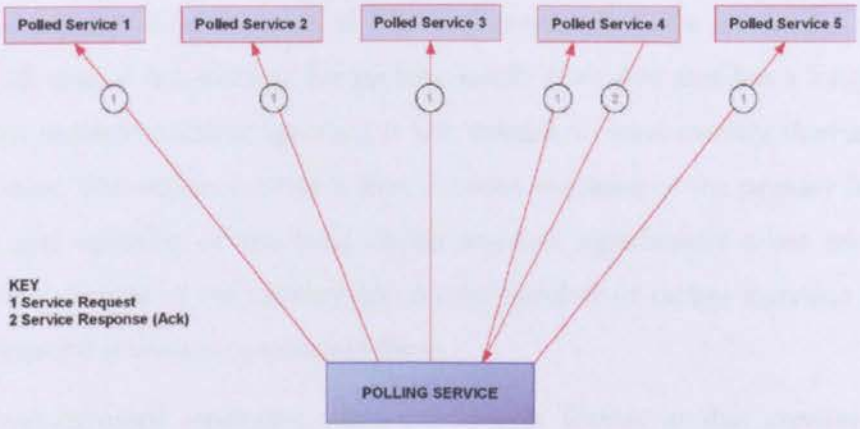

Figure 61 Operational Model of the Broadcast Active Respond Strategy

Transactional Publish Subscribe Strategy

The "Transactional Publish/Subscribe" method involves publishing a service request to a list (*see Figure 62*). The service request list is a FIFO list and any servicing entity may remove the first request from a list, however only one service may subscribe to process a single service request. Upon completion of a service request the servicing entity sends its response to the Posting service. It is possible to have multiple posting services publishing to the same service list. The posting service should have a timeout mechanism to recover if a response to a published request is not forthcoming.

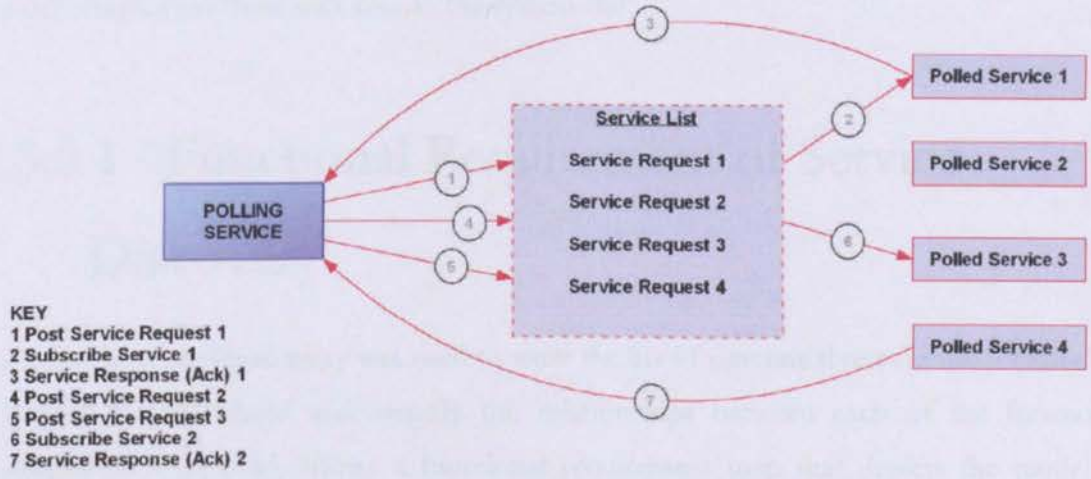

Figure 62 Operational Model of the TP Subscribe Strategy

As part of the "Transactional Publish/Subscribe" method the question arises as to where the location of a service list resides and if the list should have local copies (caches) located across the cluster rather than in a central location. The relevance of whether a local cache is

advantageous depends on the nature of the data stored within the service list. Data which has a short life span is less suitable for caching locally than data that has a longer life span. Also data that requires constant updating is less suitable to local caching than any data that has a fixed value. The reason for this is that constant updating of the primary list affects all cached lists and updating of the local caches requires significantly more resources than simply the maintenance of the primary list. As the number of caches increases so does the amount of required resources to maintain them.

All the above discussed strategies, share a common format in that services expose an availability notice to the service discovery element and running services communicate to each other using one or more of the strategies mentioned.

4.3.5 Requirement Model

The requirement model is an essential part of the investigation, as it enables us to define the functional requirements of service discovery and in questioning these functional requirements, one is able to model the quality attributes, or CTQs which provide the benchmarking indicators that drive the simulation exercises. On the one hand, functional requirements provide insight on "what the system does?" and on the other hand, the CTQs provide insights on "how well should the system do?"

4.3.5.1 Functional Requirement of Service Discovery

A requirement catalogue entry was used to write the list of functional requirements (Rose92). It helped to understand and identify the relationships between each of the functional requirements. Figure 63, shows a functional requirement map that depicts the model of service discovery problem for the domain of distributed messaging applications.

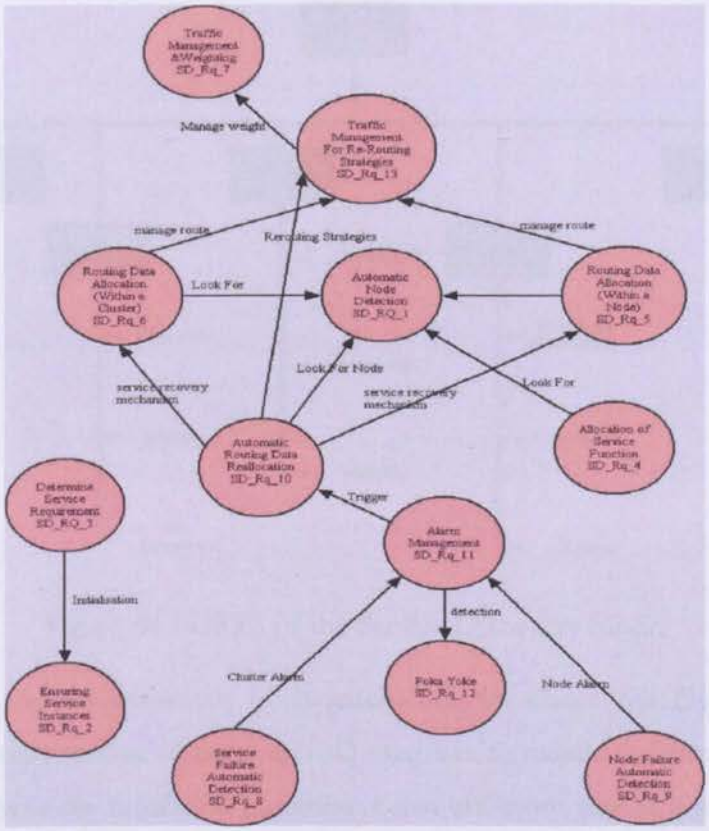

Figure 63 Functional Requirement Map of the Service Discovery Model

The purpose of a requirement map is to define the relationship between functional requirements and through the process of labelling the relationships, one is able to produce a list of functional interactivity amongst requirement. This can be easily translated into a flow chart within the blended modelling approach. In showing which requirements have the most interactivity, the map can be used as a measure of complexity. However, this is not always a good measure since some requirements may have greater operational utility to the user of the systems, yet less functional dependencies and this is not portrayed in this model. To define the user utility, designer will use the approach of operational profiling to analyse the interactivity and usage of a particular requirements with a user group.

4.3.5.2 CTQs of Service Discovery

A quality model for service discovery was built using the FURPS, as before, to organise the CTQs that we need to measure during the simulation analysis.

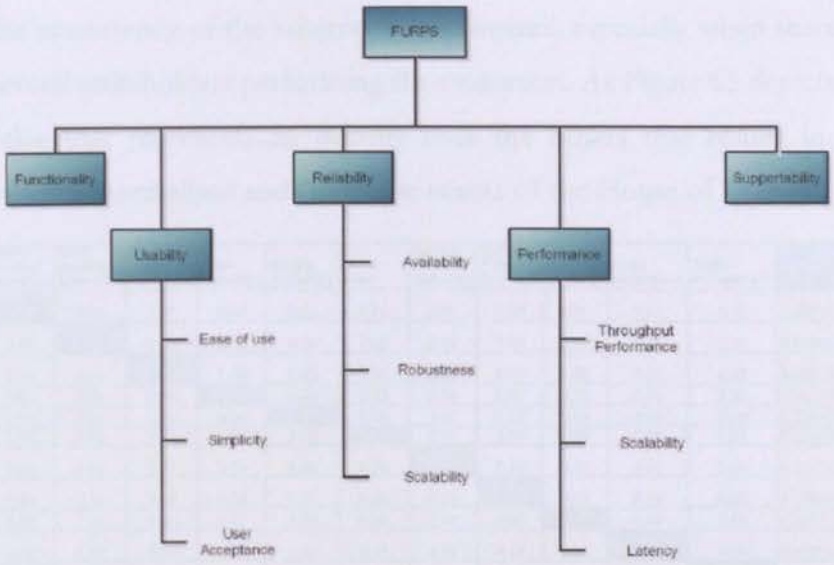

Figure 64 FURPS of the Service Discovery Model

For the analysis and measurement of requirements, we chose the HoQ and the AHP matrices. The primary motive of using a HoQ matrix is to reduce the number of ineffective simulations and focus on simulation parameters that are more stringent to the problem. By narrowing down the CTQs, q, to address the most essential features, in the equation $S = q \times (cs \times cam)$, where S is no. of simulations, q is the no. of CTQs, cs is the no. of communication styles and *cam* is the communication agreement model, we reduce the cost, time and resources needed to model and validate the specifications of service discovery. The algorithm behind the HoQ requires a value to amount the importance of a particular CTQ vis à vis a functional requirement. However, HoQ on itself does not provide a formal method to evaluate the importance of a given CTQ. Instead we complemented the HoQ matrix with another quality modelling tool called the Analytical Hierarchy Process (AHP) matrix.

4.3.5.3 The Analytical Hierarchy Process (AHP) Model

The AHP matrix has been implemented to evaluate the importance of the functional requirements against each other (*see Figure 65*). The evaluation process is subjective but carried out by expert groups and based on domain knowledge. The purpose of the AHP is

to improve the consistency of the subjective assessments, especially when there are multiple criteria and several stakeholders performing the evaluation. As Figure 65 depicts, each CTQS is given a value that represents its priority over the others that results in an order of importance which is normalised and fed to the matrix of the House of Quality.

Factor	Cluster Routing Allocation	Node Routing Allocation	Service Node Allocation	Node Detection	Service Availability	Poka Yoke	Service Failure Detection	Node Failure Detection	Re Routing	Event Notification	Traffic Management	GEOMEAN	Normalized Weight
Cluster Routing Allocation	1.00	3.00	0.33	0.50	0.20	5.00	2.00	3.00	2.00	3.00	3.00	1.4371111	0.104475
Node Routing Allocation	0.33	1.00	0.25	0.33	0.20	2.00	2.00	3.00	0.50	2.00	2.00	0.8326241	0.0605301
Service Node Allocation	3.00	4.00	1.00	2.00	3.00	7.00	3.00	4.00	2.00	5.00	4.00	3.0862045	0.2243607
Node Detection	2.00	3.00	0.50	1.00	0.20	3.00	2.00	2.00	0.33	2.00	2.00	1.2271563	0.0892119
Availability	5.00	5.00	0.33	5.00	1.00	4.00	3.00	3.00	2.00	3.00	3.00	2.5264318	0.1836116
Poka Yoke	0.20	0.50	0.14	0.33	0.25	1.00	0.17	0.20	0.25	0.50	0.33	0.2866825	0.0217122
Service Failure Detection	0.50	0.50	0.33	0.50	0.33	5.88	1.00	2.00	0.50	3.00	2.00	0.9372421	0.0681356
Node Failure Detection	0.33	0.33	0.25	0.50	0.33	5.00	0.50	1.00	0.33	3.00	2.00	0.7094517	0.0515757
Re Routing	0.50	2.00	0.50	3.03	0.50	4.00	2.00	3.03	1.00	4.00	3.00	1.6331146	0.1187241
Event Notification	0.33	0.50	0.20	0.50	0.33	0.33	2.00	0.33	0.33	1.00	0.33	0.4454951	0.0323868
Traffic Management	0.33	0.50	0.25	0.50	0.33	3.00	0.50	0.50	0.33	3.00	1.00	0.6200506	0.0450764
												13.755547	1
	0.10448	0.18159	0.07479	0.04461	0.03676	0.10866	0.13627	0.1547	0.2374	0.09716	0.1352292	12.664366	
	0.03483	0.06063	0.05609	0.02974	0.03676	0.04342	0.13627	0.1547	0.0694	0.064773	0.0901528	12.666697	
	0.31343	0.24212	0.22436	0.17842	0.55144	0.15199	0.20441	0.2063	0.2374	0.161933	0.1803056	11.820903	
	0.20895	0.18159	0.11218	0.08921	0.03676	0.06514	0.13627	0.1032	0.0392	0.064773	0.0901528	12.636867	
	0.62238	0.30265	0.07479	0.44608	0.18381	0.08685	0.20441	0.1547	0.2374	0.09716	0.1352292	13.304386	
	0.0209	0.03027	0.03206	0.02974	0.04596	0.02171	0.01158	0.0103	0.0297	0.016193	0.0160255	12.132004	
	0.06224	0.03027	0.07479	0.04461	0.06127	0.12772	0.06814	0.1032	0.0694	0.09716	0.0901528	11.871127	
	0.03483	0.02018	0.05609	0.04461	0.06127	0.10866	0.03407	0.0516	0.0392	0.09716	0.0901528	12.363669	
	0.06224	0.12106	0.11218	0.27034	0.09191	0.08685	0.13627	0.1563	0.1187	0.129546	0.1352292	11.881602	
	0.03483	0.03027	0.04487	0.04461	0.06127	0.00724	0.13627	0.0172	0.0396	0.032387	0.0160255	14.312283	
	0.03483	0.03027	0.05609	0.04461	0.06127	0.06514	0.03407	0.0258	0.0396	0.09716	0.0450764	11.843439	
	λmax =	9.15909										137.36632	
	C.I. =	-0.41721	< 0.15										

Figure 65 AHP Analysis of Requirements for the Service Discovery Model

The results of the AHP matrix is summarised into the following graphs and bringing an order of importance to the functional requirements.

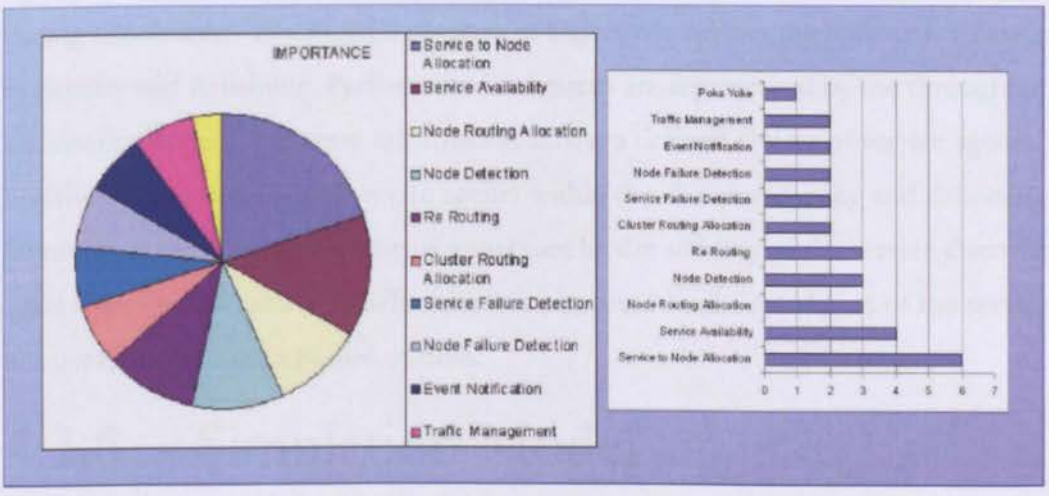

Figure 66 AHP Results of Requirements for Service Discovery Model

4.3.5.4 The House of Quality (HoQ) Analysis

The HoQ in Figure 67, enables designers to evaluate the functional requirement against the CTQs and using the order importance of the AHP as weight. The evaluation is carried out using the principle of asking how critical a particular CTQ is for a given functional requirement. The answer is represented by a metric of high, medium and low. This is then summarised by the horizontal bar chart of the HoQ matrix and in our context it shows the 3 most stringent quality attributes of service discovery, which are performance, scalability and reliability.

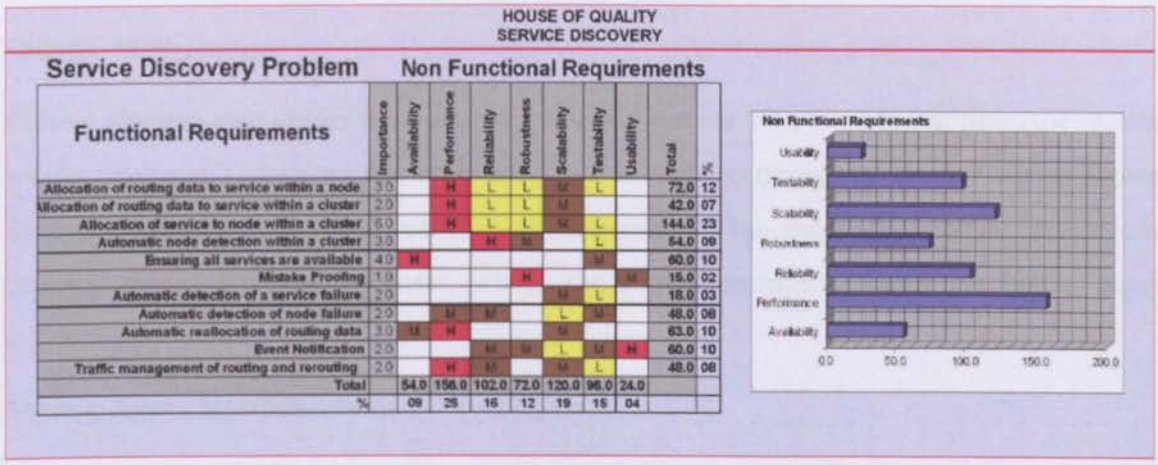

Figure 67 HoQ of Requirements for Service Discovery Model

However these quality attributes needs to become controllable so that they are measurable during simulations. The FURPS diagram in Figure 64, defines the metrics for Performance, Scalability and Reliability. Performance indicators are represented by the throughput of distributing service discovery information across a defined cluster of service agents. Scalability is the number of service agents within the cluster receiving and delivering service discovery information. Reliability is expressed by the stability of the service discovery traffic against the overall network traffic which is measured by the steadiness of the service discovery traffic over a period of time.

4.3.6 Simulation Model

As mentioned earlier, there are five communication agreement models that define the state of a cluster within a life cycle. The following sections show the characteristics of the

communication agreement models to understand the frequency of their occurrences within the life cycle of a cluster and to test each service discovery strategy (communication style) against each of the communication agreement models.

Cluster Establishment

Cluster establishment happens once in the system life cycle, at initialising time. There are several service managers and each manager is required to know about their neighbours. The simulation performs a many to many relationships between the service managers wherein all service discovery strategies compete with each other to find the best strategy for service discovery at that level.

Cluster Maintenance

Cluster Maintenance occurs periodically during the system life cycle. At run time, the master service manager communicates with the slave service managers to check their existence through distinct "*heart beats*" which is configurable. This process maintains a quorum defining the instance of a cluster, where the communication is based on one-to-many relationships.

Service Agent Establishment

Service agent establishment happens once during the system life cycle, when a cluster is established and its boundary is defined. The next operation is to identify and initiate the service agents within the cluster. The service managers communicate with the service agents, which is based on one-to-many relationships.

Service Agent Enablement

Service agent enablement happens once during the system life cycle. It concludes the final phase of initialising the service agents, after all the service agents have been established. In our scenario, the service agent enablement process will cluster all the related service agents to formulate a particular instance of a messaging gateway application. This is a many to many communication style of service discovery.

Service Agent Maintenance

Service agent maintenance occurs periodically during the system life cycle. In this state, service agents are connected and communicate to complete a typical messaging application

(e.g. VMR). During this operation, some information might be required from service agents, thus triggering service discovery, which is a many to many communication style of service discovery.

We observe that service discovery mechanisms are implemented at different states of a cluster life cycle and also that it is used at two phases of execution; 1) at initialisation time – *frequency of occurrence is once*, 2) during run time – *frequency of occurrence is periodical.*

In order to simulate the different service discovery scenarios, the high level description of the proposed communication styles are translated into a dynamic Petri Nets model.

4.3.6.1 The Broadcast Strategy

Figure 68, shows a number of configurable Places representing individual agents, where each of them has two states, ready and performing, depicted by the place *Agent.* When the simulation starts, the transition *SendMessage* is fired to broadcast discovery requests to all the clustered agents. The post condition of such flow is defined by the place *sent* which is of type *TimedMSG.* A message therefore, consists of a source and a destination. When a message is sent, the destination agents receive those messages which enable the transition *ReceiveDTMessage* to occur, firing the place *received.* On receipt, the agents process the message, shown by the transition *ProcessMessage* and acknowledges to the source which is done by firing the transition *SendAcknowledgement.* The place *ack* explains that the source agent has received an acknowledgement, i.e. the transition *ReceivedAcknowledgement* is fired. This ends the first round of broadcast strategy and another process with a different source is initiated.

Figure 68 describes the discovery traffic that occurs during a broadcast where the Petri Nets distinguish between the application traffic and the network traffic. Application traffic is any type of traffic that is not related to service discovery but can request an initiation of service discovery. An analogy is a service agent dedicated to storage and its objective is to read from and write to databases. So the reading and writing of data exchange occurs over the application traffic.

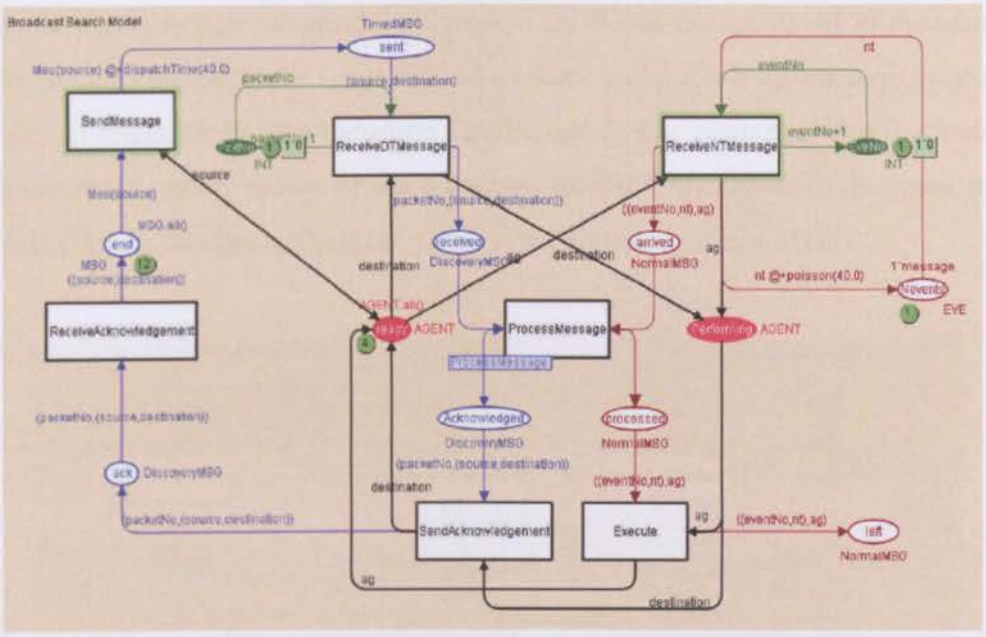

Figure 68 CPN Model of the Broadcast Strategy

In Figure 68, there is a producer of events that causes application traffic by assigning a message for agent. When a message arrives, an agent fires the transition *ReceivedNTMessage*. Consequently the place *arrived* shows a message exist and an agent needs to process the message. The transition *ProcessMessage* is triggered and the place *processed* contains a resource which is a message. The final phase of the system is the transition *Execute* that occurs when the message leaves the system. Therefore the model allows agents to handle both application traffic and discovery traffic. The impact of network traffic over application traffic provides a key indicator of throughput performance which can be observed through simulation.

4.3.6.2 The Transactional Publish Subscribe Strategy

Figure 69 shows some similarities in the design of the broadcast and the Transactional Publish Subscribe (TPS). Transitions such as *ReceiveDT/NTMessage*, *ProcessMessage* and *Execute* perform the same tasks as described in the broadcast model. However, there exist an agent pools where agents are able to subscribe to a service for publishing information on a board. This is depicted by the transition *SubscribeAgent*, which after successfully subscribing to the board, represented by the place *subscribed*, the agents enters a FIFO queue, *EnterAgentQueue*

and goes to the ready state, shown by the place *ready*. When the agents are in the ready state, three things may happen; 1) the agent may be a source and publish information on the board which will consequently fire the transition *SendMessage*; 2) The agent may be a destination and receives discovery traffic, shown by the transition *ReceiveDTMessage*, or 3) the agent may also be a handler which handles application traffic when a message event occurs.

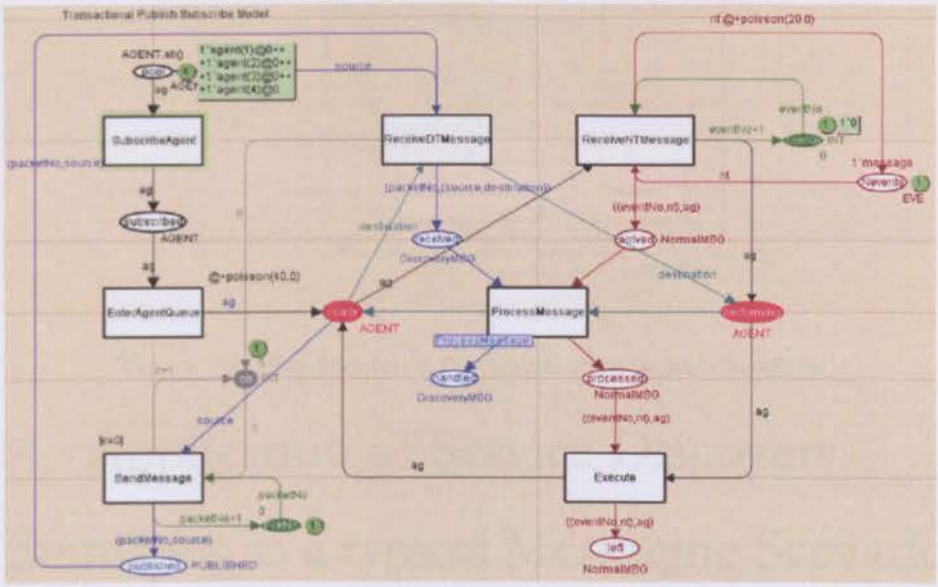

Figure 69 CPN Model of the TPS Strategy

4.3.6.3 The Exhaustive Search Strategy

The Exhaustive search model (*see Figure 70*) is again very similar to the broadcast model. The difference is that in exhaustive search, the message from the transition *SendMessage* is not publicised. Here only, one message leaves the transition *SendMessage* and one destination agent receives it, shown by transition *ReceiveDTMessage*. The source agent will not send another message until it receives an acknowledgement from the destination agent. The place *wake* & *sleep* allows an agent to send a message but it needs to wait until it receives an acknowledgement from the destination to be able to ask another agent.

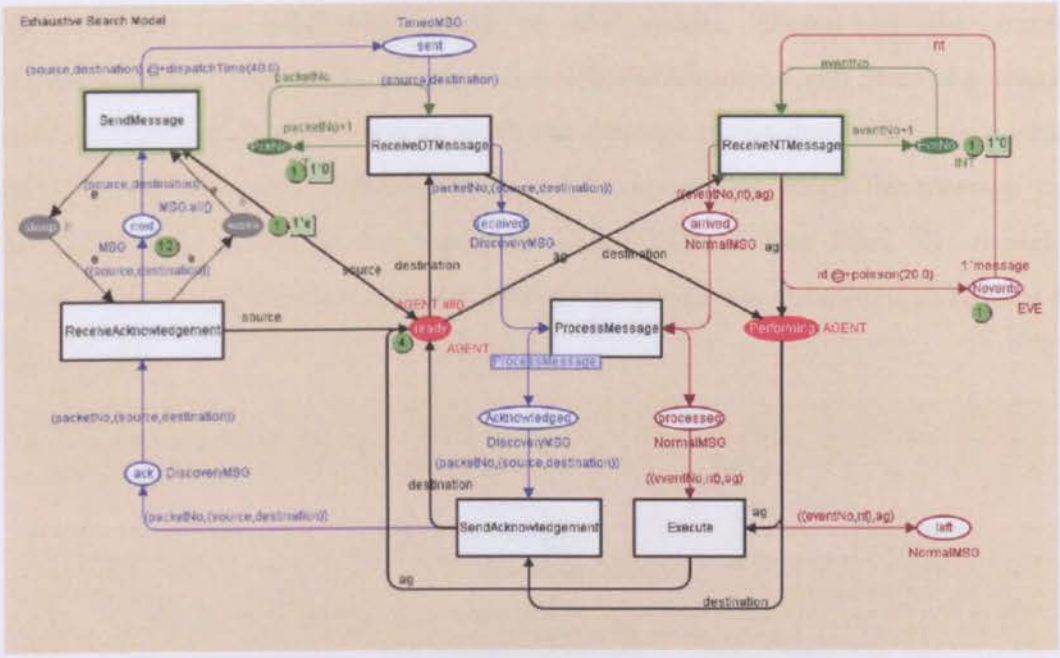

Figure 70 CPN Model of the Exhaustive Search Strategy

4.3.6.4 Application of Service Discovery Strategies to a typical Messaging Scenario

This section introduces a scenario, which attempts to position the concept of simulation and dynamic modelling into a real life situation of the VMR messaging operations. The process is called a Delivery Request which is a work flow model, designed to represent that a SMSC requires an acknowledgement that a message has been delivered to an IP based application. The objective of the simulation scenario is to find out which service discovery strategy (communication style) best conforms to the specifications of this particular work flow.

When the simulation starts, a transition *ApplicationSendMSG* sends message to the system, hence the place *arrived* indicates that a message has reached the system (*see Figure 71*). Upon arrival, the message is processed i.e. an agent is assigned to the message, resulting to the transition *ProcessMessage* to be fired. Each message has a *ticketNo* which is checked by the agent to know which SMSC the message is ought to be sent. If the agent does not know about the message, it finds it by triggering a service discovery mechanism. As mentioned there are three types of strategy, *Broadcast (all respond / active respond); Transactional Publish Subscribe; Exhaustive Search.* When an agent who knows about the existence of the message is

found, it sends an acknowledgement to the sender, shown by the transition *sendAcknowledgement*. The sender receives the acknowledgement and fires the transition *ReceiveAcknowledgement*. Next the sender sends the message to the destination agent which is depicted by the transition *SendMessageToAgent*. The agent then sends the message to the committed SMSC, shown by the transition *DeliverToSMSC*. The Petri Net model was simulated wherein each service discovery strategy proposed was submitted to the test for a delivery request.

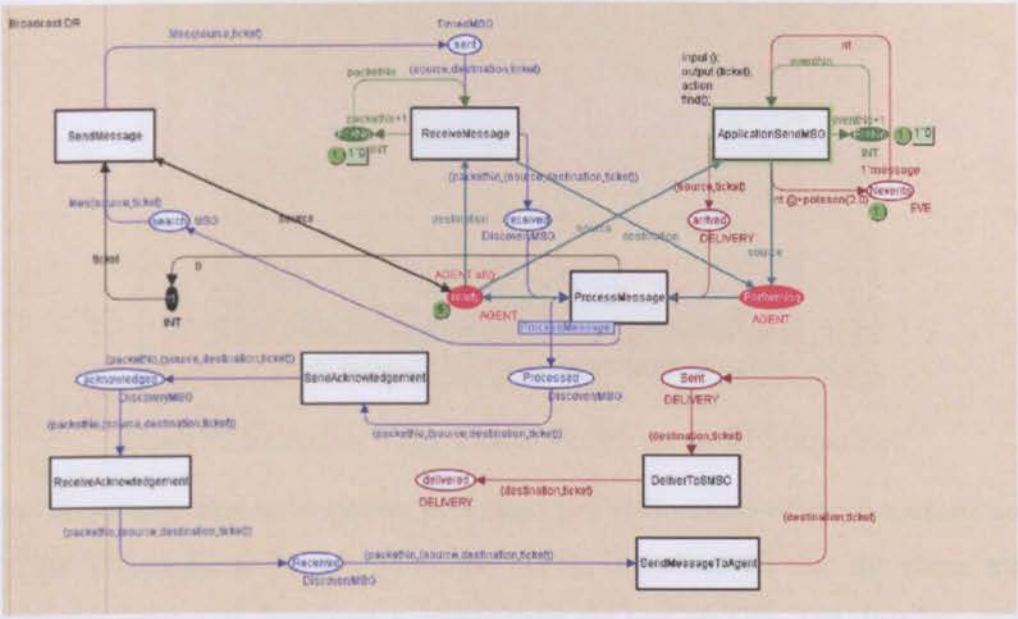

Figure 71 CPN Model of Delivery Request Model in Messaging

4.3.7 Observations

A cluster is complete after the service agent enablement state has been reached, thus all the service agents, that are required to complete a distinct messaging gateway application, are enabled. The service managers are responsible for one or more service agents. In order to establish a messaging gateway, the service managers need to find out about themselves and ask each other the type of service hosted.

We looked at the generic model of broadcasting, exhaustive search and transactional publish subscribe. Three simulations of 30,000 steps with one millisecond interval each, were executed. We change the agent population from 4 to 20 units for each simulation to address the quality attribute of Scalability. The first scenario considered was the process of

establishing a cluster, referred to as the cluster establishment. After the simulation, we tested for the normality of the observed results, then conducted a 2 sample T Test (Park07) on the service discovery message processed per unit time. We then compared which strategy has a greater mean for message processed per second, i.e. the speed for processing service discovery message relating to the quality attribute of Performance.

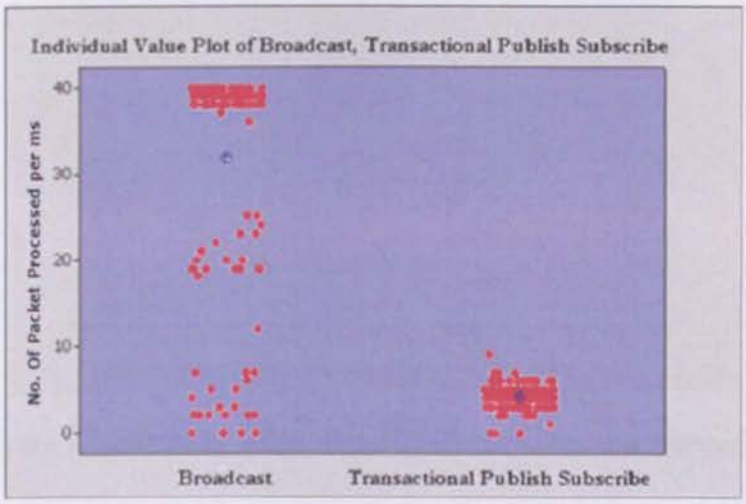

Figure 72 Individual Value Plot of Broadcast vs. TPS Strategies

We plot the service discovery packet processed per millisecond of the broadcast strategy against the Transactional Publish Subscribe (TPS). We observed that the mean message processed per second of Broadcast is bigger than the one of TPS. The distribution shows that the Broadcast strategy allowed more messages per second to be processed by destination agents rather than TPS. Hence the likelihood of Broadcast strategy finding a bigger population of agents within a period of time in a system is greater than TPS. However, the standard deviation of Broadcast against its mean is bigger than that of the TPS where the individual plots are more crowded around the mean (*see Figure 72*). Yet the distribution of TPS is steadier which shows that the overall network traffic is more stable and reliable. This is understandable since within the TPS model, there exist a FIFO queuing model allowing only one agent to process only one message for each simulation turn and if there is a burst of incoming message, the message waits in the queue's buffer instead of flooding the system. TPS provides reliable network mode, yet the drawback for such strategy is that message may pile up at the queue, and messages have to wait longer than, perhaps, the defined Service Level Agreement (SLA). The flexibility of such system lies on how elastic the FIFO queue is, i.e. dynamic adjustment of its buffer.

When compared to the Transactional Publish Subscribe strategy, Exhaustive Search really shows that very few messages are processed per second, thus exhibiting poor performance. As the number of agents scaled up, the performance dropped drastically (*see Figure 73*).

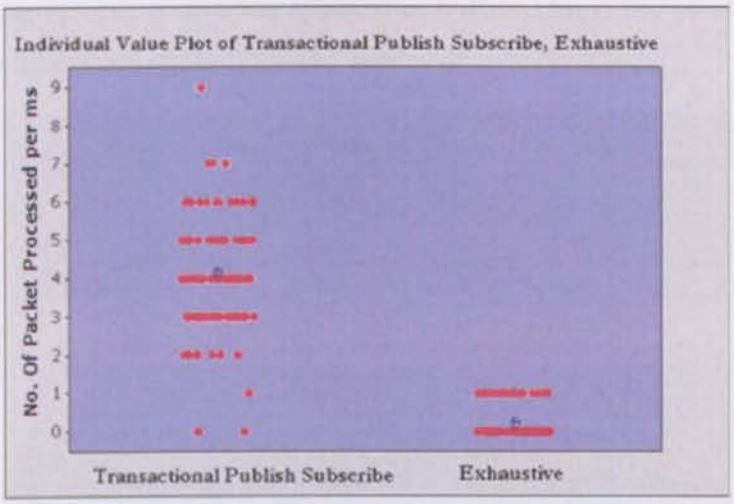

Figure 73 Individual Value Plot of TPS vs. Exhaustive Strategies

So both performance and scalability were poor for the Exhaustive Search strategy. We can understand that because in exhaustive search, when the source is looking for a destination agent, he waits idly for an acknowledgement and then search for the following agents, and so on, thus within a given time period, less discovery message is processed. However, with the exhaustive search strategy, the process to looking for a service and waiting for an answer make the strategy more robust, which addresses the quality attribute of Robustness. Should the reply from the destination agent be critical and necessary, we demonstrate that the exhaustive search strategy provides more robust solutions.

In Figure 74, the Box plot of the broadcast strategy against the transactional publish subscribe strategy, is another representation of the number of message processed per second. The area of the Box Plot for Broadcast is larger than TPS and it shows the weight of the distribution for message processed per unit time against TPS. So far, we found that the Broadcast is faster in processing service discovery message per a given unit time than TPS but TPS is faster than Exhaustive Search. What is required to understand is how the three service discovery strategies influence the application traffic of the network. When sampling data for the T Test of application traffic, we observed that there were not sufficient data points for the broadcast strategy. This is due to the fact that compared to the other two

service discovery strategies, the broadcast strategy processes far too little of the application traffic. This is logical and observable when we increased the number of agents in the CPN model to 20 agents; a broadcast strategy floods the system with service discovery messages leaving very few agents idle to process application traffic. This is the case when there is no network separation between application and service discovery traffic.

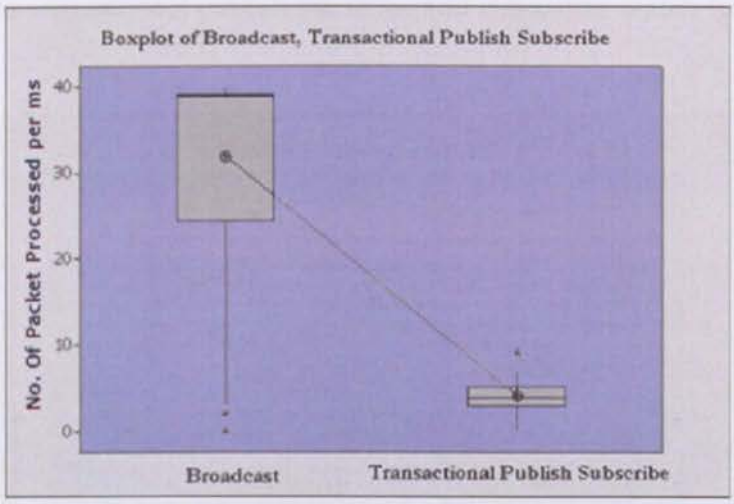

Figure 74 Box plot of Broadcast vs. TPS Strategies

Figure 75 shows a graphical visualisation of application traffic processing between the broadcast TPS strategies.

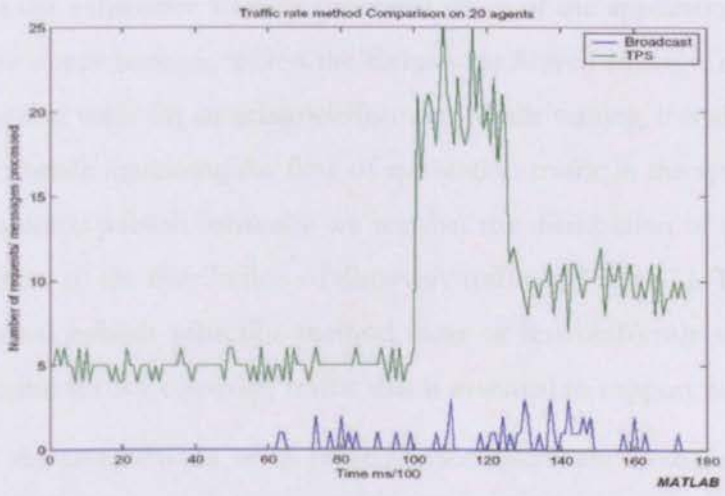

Figure 75 Throughput Performance of TPS vs. Broadcast Strategies

The result of the analysis compares the throughput performance of application traffic of two systems, one using a broadcast strategy for service discovery, and the other utilises

transactional publish subscribe (TPS) method with each system hosting 20 software agents. It can be observed, that the worst case of deprivation for application traffic is when the broadcast strategy is implemented. Using the transactional publish subscribe strategy application traffic performance was observed to be much higher. We applied a two sample T test (Park07) to compare the mean of the number of application traffic messages processed per second for the transactional publish subscribe and exhaustive search strategies (*see Figure 76*).

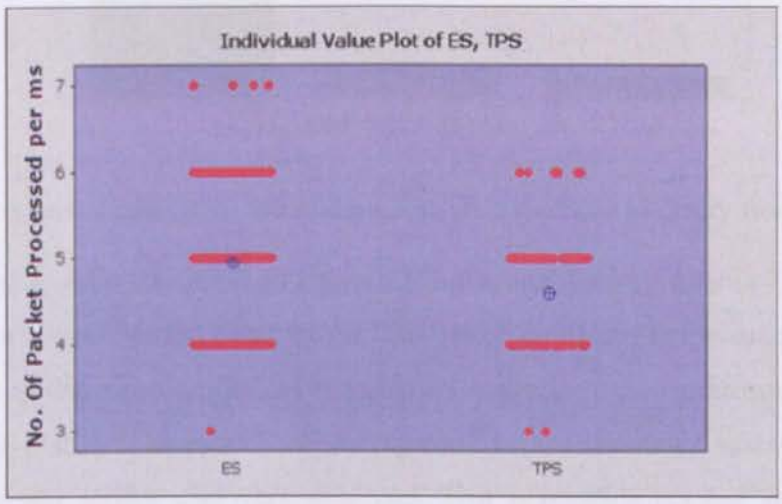

Figure 76 Individual Plot of Exhaustive Search vs. TPS Strategies

We observed that the exhaustive strategy processes more of the application traffic than the TPS strategy. This is true because, within the Exhaustive Search strategy, one agent sends a discovery message and waits for an acknowledgement. While waiting, it is idle to carry out its application traffic, hence increasing the flow of application traffic in the system. However, if we look at transactional publish subscribe we see that the distribution of application traffic in Figure 76 is similar to the distribution of discovery traffic in Figure 72. This demonstrates that the transactional publish subscribe method more or less uniformly shares the load of application traffic and service discovery traffic that is essential to support high scalability.

Figure 77 shows the comparisons of all three Service Discovery strategies applied for the single problem of Delivery Request (*see Figure 71*).

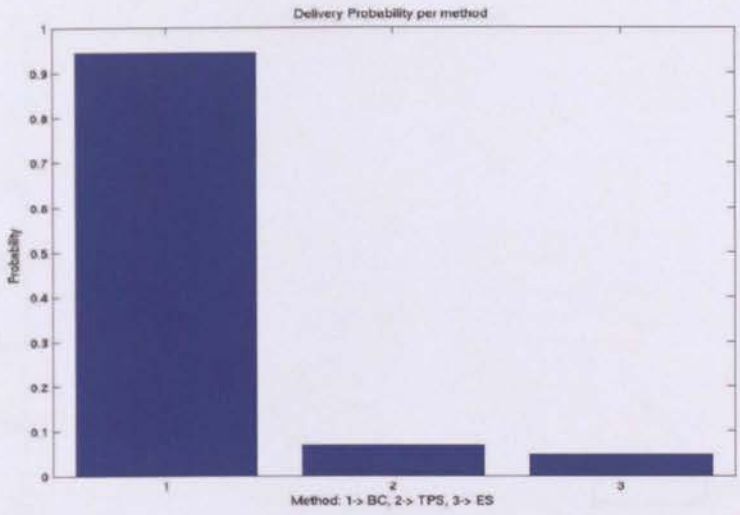

Figure 77 Broadcast, Exhaustive Search and TPS strategies in a Delivery Request Scenario

The metric used to draw the graph, in Figure 77, is the cumulated probability that a packet is delivered within 10 millisecond of its arrival. This graph compares the mean evolution of the probability for all three service discovery strategies and shows their performance. The ideal system would guarantee a mean of 1. As we can observe, the broadcast strategy is closer to 1 than the other two service discovery strategies. This is possible because the Petri Net in Figure 71 models the agents as a multi-threaded system and can run service discovery traffic and application traffic concurrently. That model makes the assumption that service discovery and application traffic run on separate networks which implies that the broadcast service discovery strategy has a higher probability of delivering a packet within a small time of its arrival on a multi-threaded model dichotomising the application traffic with the Service Discovery Traffic. However, we did that at a cost of introducing a new problem of economics, since threads consumes CPU processing time (resources).

Figure 78 shows the ratio of packet delivery over packet arrival within 10 ms and allows packet forwarding should the ratio be outside the 10 ms timeframe. This occurs when packets are not delivered in turn but wait in the system and are delivered in the next or next + n turn. The Box plot shows the performance and reliability of each strategy for delivery request. The standard deviation tells us that transactional publish subscribe is steadier for service discovery per message and the least steady is broadcast.

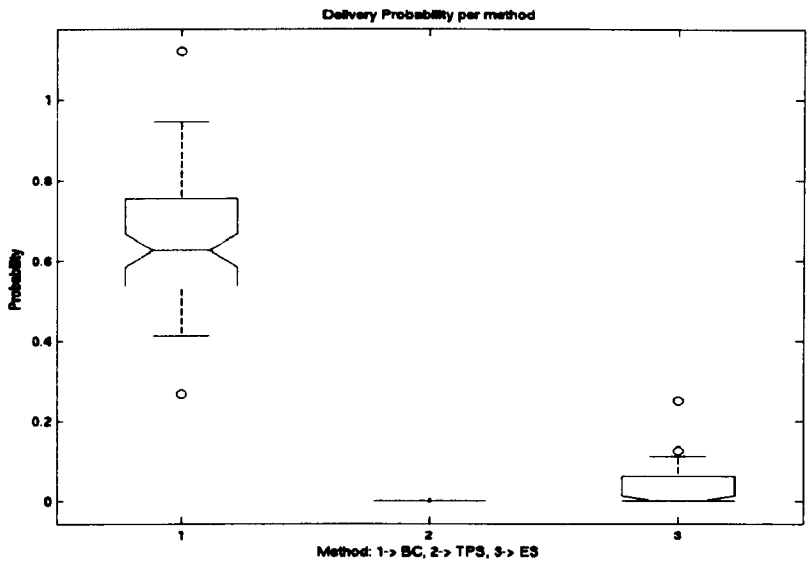

Figure 78 Box plot of Broadcast, Exhaustive Search and TPS strategies

So far what we have observed is that:

- Broadcast is faster than the other two strategies to discover new services
- Broadcast deprives the system from application traffic since the system is overwhelmed with discovery traffic.
- TPS more or less shares the application traffic and discovery traffic uniformly.
- TPS allows the system to run at a steady state due to the FIFO queue.
- ES allows more application traffic in the system and minimises discovery traffic.

On the grounds of our finding, we see that one service discovery strategy alone cannot conform to all requirements of the communication agreement models within a given distributed messaging systems. At each phase of the communication life cycle a blended service discovery model is required as summarised in Table 4.

	Cluster Establishment	Cluster Maintenance	Service Agent Establishment	Service Agent Enablement	Service Agent Maintenance
Exhaustive Search				Reliability	
Broadcast All Respond	Performance				
Broadcast Active Respond			Performance		
Transactional Publish Subscribe		Stability – Steady State Management of Scalable nodes			Stability – Steady State Scalability

Table 4 Observational Matrix of the Service Discovery Model

4.4 Conclusion

In this chapter we started a detailed analysis and simulation of a real life messaging systems (VMR) in order to assess the level of QoS required in messaging and to investigate the potential use of distributed systems within a messaging environment. Within such environments the medium of communication plays a vital role in terms of interconnectivity where queues are the most commonly used data structure. Thus, we modelled the VMR, to assess the behaviour of several concurrent interconnected queues using CPN by handling the key parameters of the queueing model also known as the Critical Parameter Management.

We started the design by mapping the relationships between the functional requirements and the non-functional requirements (CTQs) in order to have a clear definition of the quality attributes which do show impact on the system's functions and may conflict among themselves. These functionalities are translated into Petri nets where we were able to compare the arrival rate of messages against the departure rate, hence devising the transit time of a single message within the system. This type of knowledge is vital, especially in the Telecoms when Service Level Agreements (SLA) are being signed. We also simulated the system to assess the buffer population of the queues given a fixed throughput of messages and from these statistics, designers can plan for the capacity of the queues by adjusting the throughput parameters.

The next step was to migrate the VMR agent model onto a distributed architecture and simulate the various communication styles within the distributed systems, leading to the concept of service discovery. We demonstrated how these strategies work across distributed nodes and how they influence the choice of communication styles. The results of the experiments demonstrated that one service discovery strategy alone cannot provide full compliance to the overall system requirements. This means that at various stages of system life cycle, different types of service discovery strategies have to be implemented in order to manage the nodes. To know which strategy to use, we have to look at the characteristics and CTQs which are essential for that particular communication agreement model. The example that was considered shows that to establish a cluster of nodes, one of the CTQs is performance and one of the characteristics is the occurrence of that process. In this case the observational results illustrate that the Broadcast strategy matches the criteria and should be implemented as far as performance is concerned. Moreover, we also showed that the Broadcast strategy overwhelms the network traffic and is not recommended during high volume of application traffic. Since establishing a cluster happens only once during the system life cycle, prior to any application traffic, the strategy conforms to the characteristic of the communication agreement model. We have shown that this type of knowledge can only be obtained through dynamic modelling techniques and indeed such knowledge has been acquired from the simulation of the other common communication.

5 EVOLUTION WITHIN DISTRIBUTED MESSAGING SYSTEMS

It is not the strongest of the species that survives, nor the most intelligent, but the one most responsive to change.

Charles Darwin

5.1 Overview

The aspect of evolution to software solutions within the Telecoms industry is significant in terms of survivability and viability. The deployment of computerised systems that evolve within their dynamic environment requires building software that are autonomous and distributed by nature (Leh80b). This chapter explains how we tackled the problems of evolution and adaptability by proposing a novel distributed messaging system based on four of Lehman's laws of evolution (*see chapter 2*), i.e. a) *Increasing Complexity*; b) *Continuing Growth*; c) *Decline Quality*; and d) *Feedback System* . The laws relating to decline in quality and feedback systems have been integrated into the very fabric of the proposed distributed messaging system using the new concept of Run Time Quality Assurance (RTQA). This has been achieved by developing an algorithm, referred to as the BipRyt algorithm, which addresses the problem of quality deterioration as the system scales in performance. The algorithm illustrates the use of intelligent feedback mechanisms to enable the system to adapt to its dynamic environment. The BipRyt algorithm is fundamentally a multi-purpose decision making mechanism for the management, distribution, control and optimisation of computational resources (e.g. number of processing nodes, CPUs, queues and communication channels, memory etc).

Historically, several works have been carried out in the domain of decision making mechanisms for distributed systems, which employed many strategies including: 1) a mutual

exclusion protocol, wherein a node requests permission from all other nodes to enter its critical section (Lync96); 2) a consensus strategy, where nodes have to agree upon certain issues (Dole82); 3) a leader election method, where nodes are selected to coordinate certain tasks; 4) a sender or receiver initiated protocol, typically designed for distributed scheduling, where nodes advertise their need/desire to schedule jobs and decide based on the response to the advertisement (Sing94) ; 5) a Byzantine protocol, where a faulty node is detected based on recursive message passing (*announcing the status*) and decision making (Lamp83, Awer02); and 6) a distributed deadlock detection mechanism, where messages are sent to nodes that hold resources in an attempt to break a deadlock (Tane95). The reader should note that most of these decision making mechanism are based on **single quality attribute**, for instance: reliable scheduling, faulty node detection, deadlock exclusion, performance such as response time etc. However, these decision making strategies preserve their single quality attribute, regardless of its impact on other attributes, for instance, by preserving reliability (break deadlock), performance will be degraded and vice versa.

The approach we are proposing is designed to take into account many quality attributes while preserving the overall quality of the system by continuously assessing the impact of the attributes against each other e.g. performance vs. robustness, manageability vs. scalability etc. The BipRyt algorithm establishes its decisions on a combination of quality attributes, referred to as the quality model. As mentioned earlier, the algorithm is implemented on a new concept called Run Time Quality Assurance where a system is aware of certain quality attributes ensuring that, at run time, these qualities are preserved at any time and any cost. In the domain of distributed messaging systems, the BipRyt algorithm monitors the behaviour of dispersed nodes by gathering data and building up a history of resource consumption.

The algorithm associates a resource to each quality attributes with diverse economical values, which are established on the price-based model (*see chapter 2*). But we extended further on this particular economic model to include the aspect of utility to the quality attributes of the system. These attributes are evaluated by both the system at run time and the user (human) at initialisation time. The economic model integrated into the BipRyt algorithm is more advanced than the conventional style of cost function, since it is based on a multi-criteria assessment of the resources, i.e. multiple quality attributes. In order to confirm such a statement, we explored four aspects of resource management:

1. Analysis and measurement of demand for resources
2. Analysis and measurement of supply of resources
3. Analysis and supervision of resource consumption – scarcity
4. Analysis and organisation of resource distribution – load balancing and intelligent routing

The problems that the BipRyt algorithm addresses are related to the problem of economics, which is predominantly the management and distribution of scarce resources within a distributed system. Now in applying the economic model into messaging problem domain, the theory of supply and demand describes how the system energy index (SEI) of resource consumption, such as such as 1) CPU usage; 2) memory usage 3) number of threads and 4) number of file descriptors running (I/O), and other user defined quality metrics, are being consumed. The SEI is examined during the processing of a measurable quantity of uniquely identifiable message(s) and it varies as a result of a balance between resource availability (supply) and the requirement of those with *purchasing* power (demand).

5.2 The BipRyt Algorithm

The BipRyt algorithm provides a new decision making mechanism based on 1) availability of computational resources; 2) associated rules of usage and 3) defined rules for a specific user, group of users or the system as a whole.

The algorithm allows for local and global optimisation of resources during the system life cycle. This is achieved through a rule-based system where rules can be local (i.e. attached to a **user** and therefore are considered as local constraints) or global, attached to the system as a whole (global constraints). Part of the algorithm's responsibility is to manage conflicts among the rules, focussing on the problems of racing condition and resource starvation, hence providing a balance between the two axes. The BipRyt algorithm derives and compiles directives based on the rules, ultimately acting as a mediator.

5.2.1 Quality Model Awareness

According to Khaddaj et al in their work on the evaluation of software quality factors in very large systems (Kha04), quality is a multidimensional construct reflects in a quality model, where each parameter in the model defines a quality dimension. Khaddaj et al proposes an approach in which global and locally defined quality criteria can be considered and individual quality views can be combined where **conflicts** can be handled. The BipRyt algorithm implements the concept of Quality Modelling with the objective of understanding how quality attributes complement or conflict with each other. The quality model packages the attributes, (also known as the CTQs), that were defined at the requirement stage and embeds these CTQs into the system. This results in the system being aware of its resource depletion against the CTQs, which are measured at run time.

Unlike classical decision making strategies which are based on simple or single quality attributes such as Cisco's weighted round robin and response time load balancing mechanisms (Cis04), the BipRyt algorithm takes into account a series of quality attributes (multi-criteria) compiled as a quality model which are representative and adaptable to the user's needs. The concept of Quality Modelling within the BipRyt algorithm is based on the following:

- Decisions of enforcement
- Authorisation
- Resource management
- Any type of activity based on the QoS (Quality of Service) expected from the system

The quality model is built by appointing a value that represents a weight to system elements, which are CTQs that one wants to preserve during system execution. The CTQs characteristics are compiled as a Prioritisation Model, which is a set of matrices. Thus, depending on the weight, the heaviest CTQ has highest priority. This weight represents the importance of the CTQs which can be modelled by the House of Quality (HoQ) matrix. Using Saaty Calculation of the AHP (saat80), the mechanism then calculates the Eigen values or more exactly the Eigen vector from these matrices. As a result the Eigen values become the directive to follow when taking decisions. In ensuring that quality is maintain at specified

level, BipRyt resolves the problem of the 7th law of Lehman, "Decline of quality". In Brief, the BipRyt algorithm was designed primarily to take into account the operational environment (through analysing the CTQs of the system, i.e. the Eigen values) and steer the system by adjusting its parameters to conform to the environment at run time, thus preserving the quality of the system as it scales.

5.2.2 BipRyt's Practice to Lean

Taichi Ohno (Ohno88) is considered the inventor of the Toyota manufacturing model and one of the first to introduce Lean principles in manufacturing including: reducing waste, promoting concurrent development, just in time and the ability to make changes "late in the cycle". In the late 40's Toyota wanted to produce quality cars at the same price of the American mass-market producers, but the Japanese market was not big enough to allow for similar economies of scale. Toyota then re-engineered the manufacturing process following the fundamental Lean principle: eliminate waste, as shown in Figure 79.

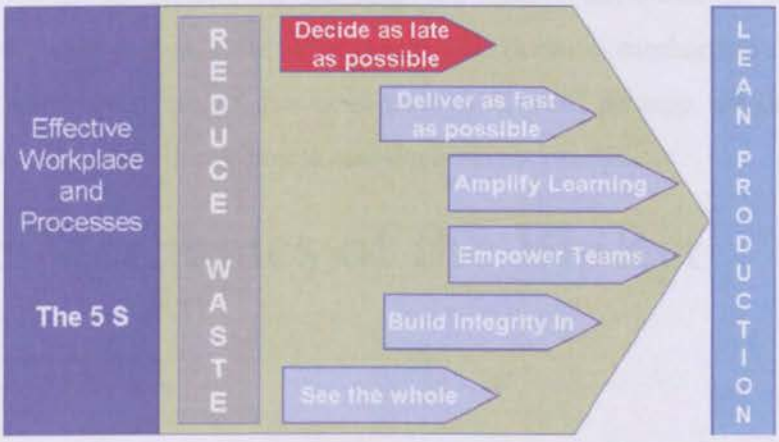

Figure 79 Lean Principles

The Japanese car industry (Toyota, Honda.) in following years became highly competitive by improving most of their manufacturing processes applying lean principles. In the context of software, the Agile Software development often claims to be the application of Lean principles to Software Engineering (Ohno88).

The BipRyt algorithm proposes a novel concept of implementing tools and techniques, which are normally applied at the requirement phase of software development process, into the software application for the purpose of Run Time Quality Assurance (RTQA). This

method aligns with the concept of "***Decide as late as possible***" which is one of the fundamental principles of the Lean Methodology which represents a paradigm shift in the approach to software design. By virtue of illustrating the shift, in classical software engineering, quality management and assurance are considered as **managerial tasks** of the development process. These tasks are usually carried out at the process level (the system that develops system). The identification of CTQs and quality modelling are activities carried out as part of the requirements analysis and design phases. However, since quality changes from perception to perception, these CTQs are relative to each user, wherein they obey to prioritisation rules and may be conflictual.

"***Decide as late as possible***" is a principle of Lean that has been adopted because of uncertainties and continuous changes (mainly due to the continuous changes in customer requirements (VOC) and the dynamics of the market) are part of the development process. If decisions are delayed the chances of incorporating all available facts and information in your decision are much higher. The BipRyt algorithm incorporates this fundamental 5th principle of lean within system, by allowing the quality attributes to be injected and configured by the user at Run Time. This means that decision mechanism of requirements (undertaken at requirement phase) has been pushed forward through the development life cycle, into the system at the development and deployment phases.

5.3 The Mechanics of the BipRyt Algorithm

A distributed messaging system is a typical socio-economic system (Bai06). There are consumers, producers or providers and scarce resources. Economic models provide several interesting contributions to resource sharing mechanism. Firstly for the last 200 years, economists have been formulating prediction models and tools for limiting the complexity by decentralising the control of resources. These are sets of mathematical models that can yield several new insights into resource sharing problems. Donald et Al postulates in (Ferg96), that software economic decentralisation is provided by the fact that economic models consist of software agents, which selfishly attempt to achieve their goals. In their research they identified two types of agents (suppliers and consumers). A consumer attempts

to optimize its individual performance criteria by obtaining the resources it requires, and is not concerned with system-wide performance. A supplier allocates its individual resources to consumers. A supplier's sole goal is to optimize its individual satisfaction (profit) derived from its choice of resource allocations to consumers. The objectives of their studies describe the economic models and their use in controlling access to resources in large and complex distributed computer systems. The authors present some of the applications of economic models specially micro-economic and game models to several resources allocation problems in computer networks and distributed systems. Several papers apply versions of this model to decentralized resource allocation in computer systems (Ferg89, Ferg90a, Ferg90b, Ferg93).

However, we found out that these studies do not consider the aspect of non-functional requirements from the users' perspectives, into their economic models. They mainly apply an economic model of resource management to default "mechanical" attributes such as computational jobs, CPU time, memory cache, disks I/O, bandwidth and communication resources. These methods do not include the user's or customer's point of view to validate the importance and utility of these attributes. The BipRyt algorithm differs in the sense that a quality model is designed and validated by the users of the system. The quality model is employed by the algorithm to 1) maintain a quality of service and 2) make decisions on workload distribution and optimisation of resources. In essence, the BipRyt algorithm preserves the quality requirement of the users of the system. We looked at the worked carried out in (Kha05), a new approach proposed by Khaddaj and Horgan, to quality modelling which seeks to combine several quality modelling approaches, whilst resolving conflicts of opinions, so that quality measurement can be both tailored to a local environment and potentially can be compared across projects. The approach provides the framework to the user of a system to consistently weigh and prioritise the CTQs of a given system. The end results from the priority model are used by the House of Quality (HoQ) enabling the process of CTQ flow down. The HoQ process is repeated through several levels of granularity until the quality attributes are measurable.

These measurable parts can then be used to populate a configurable table called which results in building an Eigen vector. The users of a system configure the AHP by prioritising the quality attributes (according to their point of views). In this case a priority list can be

created that states which quality attribute are most important and conflicting for the system. This process is carried out at the initialisation time of the system followed by the application of the AHP, at run time, for decision making.

To understand the BipRyt algorithm there are 5 basic principles:

- All software components or software agents consumes resources during execution.
- All resources have metrics; they can be measured and are considered to have quality attributes (CTQs), e.g. *resource = CPU, CTQ = CPU time.*
- CTQs can be assigned a numerical value called energy level.
- The health of the system depends on the energy levels which have to be measured and controlled.
- The energy levels decrease as the resources are consumed and increase when resources are released.

To illustrate how the working of the algorithm, we introduce two metaphors, namely Perceiver and Decider. The Perceiver collects snapshots of information on the energy levels of each resource that the software agents consume at periodic intervals. The Perceiver, hence, builds up a history of energy levels for each CTQ which is then fed to the Decider. The Decider performs some statistical analysis over the recorded quality attributes and cross references the quality model with the priority attributed by the user of the system (human fabricated AHP). Depending on the results, the Decider decides which agent is the healthiest to handle or process more information in the system. The objective of the Decider is to preserve the energy levels per resources as prescribed by the user.

5.3.1 Perceiver

Consider a system wherein software agents receive data to process and to do so consume a defined number of resources. Each agent has a list of resources and each resource has one quality attribute, e.g. resource = CPU, CTQ = CPU time. The quality attributes are represented in terms of energy levels which are held in quality containers. These containers have numerical values which determine how much energy that particular agent has for a particular quality attribute (*see Figure 80*)

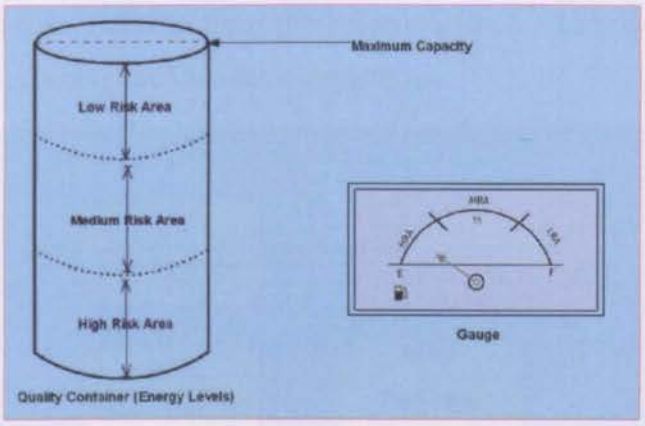

Figure 80 Illustrative Model of the Perceiver within each agent

Each CTQ, such as CPU time, is allocated with a quality container. A quality container is partitioned in three distinct areas. There is the Low Risk Area (LRA), Medium Risk Area (MRA) and the High Risk Area (HRA). HRA means that the value of a particular CTQ has dropped to a level that constitute of a great risk for the system whereas LRA means that the value of a given CTQ has dropped to a level that shows no bigger risk. The objective is to avoid the HRA region though intelligent management and distribution of work load across the quality attributes.

The energy level (value of a given CTQ) rises and drops along the three predefined areas, at system run time. The Perceiver has the ability to monitor the energy levels of the quality container which is represented through a gauge system, connected to each CTQ. When data are being distributed to a system and handled by software agents, energy is consumed and work is done by the agents to process the data. When energy is consumed, the indicator on the gauge moves towards the HRA (right). However, as agents complete the message processing, the energy levels increases and added to the container. This will result to the indicator moving towards the LRA (left). The Perceiver will record the indicator reading per processed data. When the Decider requests the information, the Perceiver sends a history of the indicator values, this builds up a feedback system based on the usage of CTQs.

5.3.2 Decider

The decider is the decision making module of the BipRyt algorithm which continuously communicates with the Perceiver. The Decider is able to perform some analytical

calculations over the data, gathered from the Perceiver, upon which decisions are made. A state chart diagram describing the Decider is depicted in

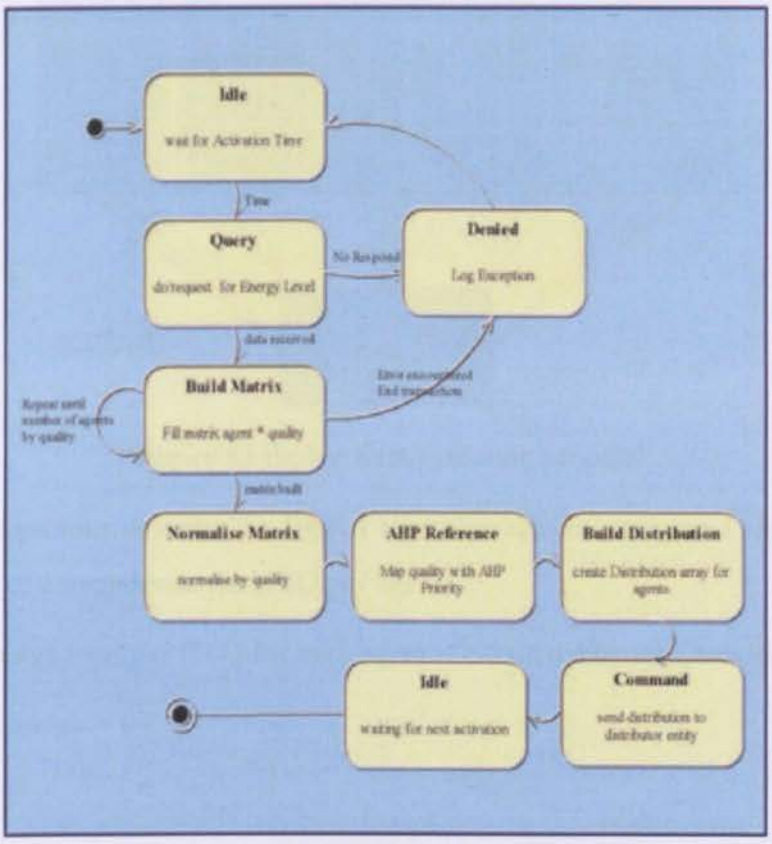

Figure 81 State Diagram of the Decider

At the initial stage, the Decider is idle and waits for activation. When activated, the Decider broadcasts a request to each Perceiver, which in return sends back a list of energy levels per CTQ per software agent. This process requires a Perceiver to take a number of energy level, from the indicator reading of the gauge and push them in a list for each CTQ per software agent, (*see Figure 82*).

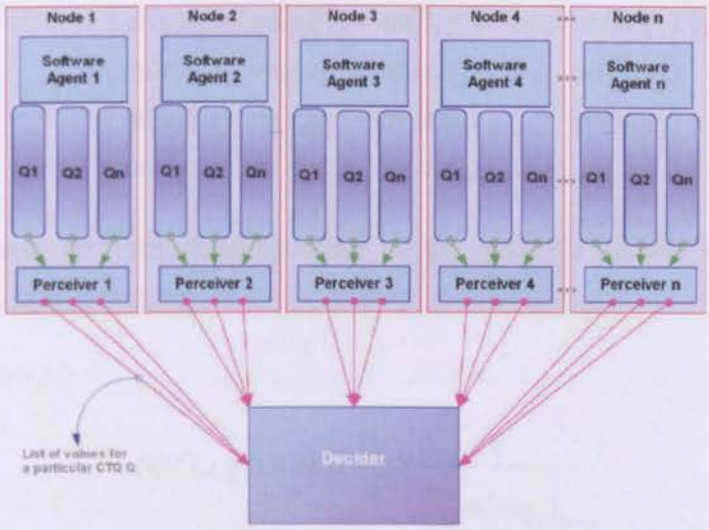

Figure 82 BipRyt data gathering process

The following equations describe the BipRyt algorithm which is triggered when the Decider receives the list of energy levels per CTQ per agent.

The mean of energy level per CTQ for each agent is calculated by the Decider.

For each Agent a_i where $i > 0$

{

For each CTQ q_j where $j > 0$

{

$$mean\ m(a_i, q_j) = \frac{1}{n} \times \sum_{k=1}^{n} V_k \quad where\ \mathbf{V_k}\ is\ the\ energy\ level\ of\ a\ CTQ\ and\ n\ is\ the\ no.\ of\ energy\ levels$$

}

}

The Decider builds a mean matrix (MM) of agent by quality and populates the array with values from the list.

$$MM = \begin{bmatrix} m(a_1q_1) & m(a_1q_2) & m(a_1q_3) & \dots & m(a_1q_j) \\ m(a_2q_1) & m(a_2q_2) & m(a_2q_3) & \dots & m(a_2q_j) \\ m(a_3q_1) & m(a_3q_2) & m(a_3q_3) & \dots & m(a_3q_j) \\ m(a_iq_1) & m(a_iq_2) & m(a_iq_3) & \dots & m(a_iq_j) \end{bmatrix}$$

The matrix can now be normalised by order of the quality attributes or CTQs into the Normalised Mean Matrix (NMM).

For each CTQ q_j where $j > 0$

$$\mathrm{NM}(a_iq_j) = m(a_iq_j) \times \frac{1}{\sum_{k=1}^{i} m(a_iq_j)_k}$$

$$NMM = \begin{bmatrix} NM(a_1q_1) & NM(a_1q_2) & NM(a_1q_3) & \dots & NM(a_1q_j) \\ NM(a_2q_1) & NM(a_2q_2) & NM(a_2q_3) & \dots & NM(a_2q_j) \\ NM(a_3q_1) & NM(a_3q_2) & NM(a_3q_3) & \dots & NM(a_3q_j) \\ NM(a_iq_1) & NM(a_iq_2) & NM(a_iq_3) & \dots & NM(a_iq_j) \end{bmatrix}$$

In order to know which quality is most important for the system, the Decider consults the human fabricated AHP table. This table is configurable by the user which defines his/her requisite. Hence, within the BipRyt operations, each normalised value of each quality for the agents is multiplied by the corresponding quality value, as set in the AHP.

$$AHP = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_j \end{bmatrix}$$

The new values of quality attributes are added together for each agent. The sum shows the distribution load of an agent.

$$\frac{\sum_{(i,j)}^{j} (NM(a_iq_j) \times AHP(q_j))}{nq}$$

Due to the fact that the distribution is calculated, based on the priorities of the users and on the energy consumption of the system, the BipRyt algorithm maximises the opportunity for

a system to conform to the user's desires or needs in terms of non-functional or quality capabilities.

5.3.3 The Architecture Model

In this section we explain the generic architectural model that employs the BipRyt algorithm that can be attached to any distributed system. The BipRyt algorithm will manage the behaviour of the system entities (for instance, the software agents) and provides a generic decision making mechanism. Signals (energy levels) from the software agents (Perceivers) will be recorded by the Decider and statistically evaluated. A Statistical Analyser is the component that provides these evaluations using the HoQ and AHP. Figure 83 illustrates the generic model and explains the various components of the architecture.

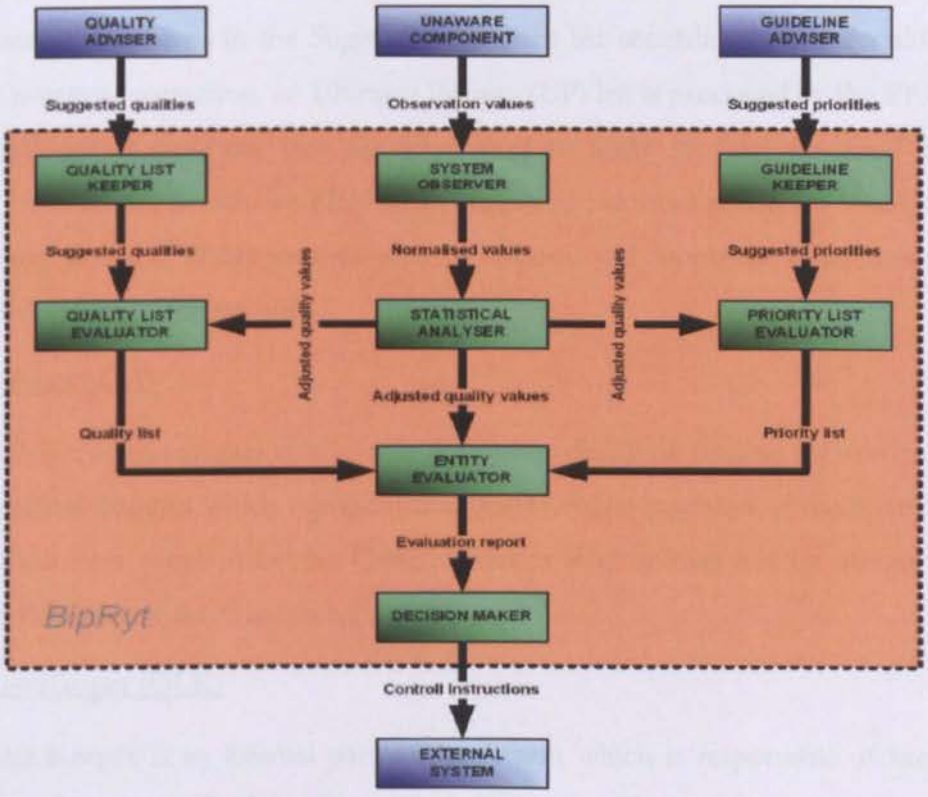

Figure 83 Architectural Model of the BipRyt System

Guideline Adviser (GA)

Guideline Adviser is an external entity to the system that feeds the suggested priority distribution values to the algorithm. The suggested priorities is the list of directives from

Guideline Adviser that represent its wishes and vision on how the algorithm should perform and what Quality attributes needs to be preserved. This in essence represents the priority values per CTQ as defined by the user requirements in the form of an AHP matrix.

Guideline Keeper (GK)

Guideline Keeper is an internal part of the algorithm which is responsible for keeping and maintaining the directives that were received from the Guideline Adviser. Directives are fed to the Keeper in the form of Suggested Priorities which are provided to any requesting part of the algorithm.

Priority Evaluator / Modifier (PEM)

The purpose of PEM is to query Suggested Priorities from the Guideline Keeper and to evaluate them for correctness, completeness and validity. It is also the purpose of PEM to modify some or all items in the Suggested Priorities list according to the decisions made. After any potential correction, an Ultimate Priority (UP) list is produced by the PEM. UP is the resulting set of priorities that algorithm uses to make its final advices / decisions. Ultimate Priorities are derived by PEM from Suggested Priorities which are based on system observations. It is also PEMs responsibility to respond and supply the evaluation results to any requesting part of the system.

Quality Adviser (QA)

Quality Adviser is an external entity, with the responsibility of feeding the system with the initial suggested qualities which represent the Quality Advisers vision of the system, (quality model). Each item supplied by the Quality Adviser may or may not be accompanied by Suggested Priorities of the Guideline Advisors.

Quality List Keeper (QLK)

Quality List Keeper is an internal part of the system which is responsible of keeping and maintaining Suggested Qualities list supplied by the Quality Adviser. It is also the responsibility of the Quality List Keeper to supply its list of Suggested Qualities to any requesting part of the system.

Quality List Evaluator / Modifier (QLEM)

The purpose of QLEM is to query the Suggested Quality lists and to evaluate them for correctness, completeness and validity. It is also the purpose of QLEM to modify some or all items in Suggested Qualities which can be modified according to its own based on observed values. It is also QLEMs responsibility to respond and supply the evaluation results to any requesting part of the system. Ultimate Quality Attributes are the resulting set of qualities that the system uses to make its final advices, derived from the Suggested Qualities.

Unaware Component / Agent (UCA)

Unaware Component (agent) is another external entity which belongs to a system that the algorithm is making decisions for. Unaware Component responds to requests regarding its current state in terms of quality usage or energy levels of CTQs. Unaware components produce Observation Value lists of qualities and report their usage to the system. Each item in the Observation Value list may or may not be accompanied by an entry in the Suggested Qualities provided by Quality Adviser.

System Observer (SO)

System Observer is an internal part of the system with the purpose of accepting and requesting Observed Values from one or more Unaware Components. It is also the purpose of the System Observer to respond with its current list of items to any request from internal components. System Observer also performs normalisation of data gathered which allows operations on balanced data, thus any further comparisons are performed on equally scaled values. Then the Normal Quality Values are supplied by the System Observer to any component of the system that requests them and used for further evaluation of external system behaviour.

Statistical Analyser (SA)

Statistical Analyser requests Observation Values from System Observer and performs statistical / historical analysis (such as mean and median) on the data retrieved. Based on the results of such analysis, the Statistical Analyser adjusts Observation Values in such a way that they are better tuned to represent current (or future) consumption of quality attributes. It is responsibility of Statistical Analyser to provide such observations to any other part of the

system. The Adjusted Quality Values are derived by the Statistical Analyser from Normal Quality Values representing the algorithm view of the external system and are used for any further calculation while making decision.

Entity Evaluator (EE)

Entity Evaluator performs evaluation of the behaviour of the external system and makes the decisions which are used to control their behaviours. The evaluation is carried out using Ultimate Priorities, Ultimate Quality Attributes and Adjusted Quality Value lists as arguments. The combination and interaction between these items are dictated by the BipRyt algorithm inside the Entity Evaluator component. The output of the Entity Evaluator is an evaluation report.

Decision Maker (DM)

Decision Maker receives the evaluation report from the Entity Evaluator. The output from the Decision Maker is a list of directives (Control Instructions) that are used to regulate the behaviour of the external system.

System Controller (SC)

System Controller is an external component to the system that receives the directives in the form of Control Instructions from the Decision Maker. Based on these suggestions, the System Controller changes the behaviour of the system accordingly.

5.4 Implementation of the BipRyt Algorithm within the Distributed Messaging System

The BipRyt Algorithm empowers the system with some decision making capabilities, which leads to the concept of autonomy, a core aspect of evolution (*see chapter 2*). The BipRyt algorithm has been integrated to distinct parts of the proposed distributed messaging system. One particular area of application is the management of traffic flow of the InfiniBand network by controlling the virtual lanes (*see chapter 6*). Another area, which is the topic of this

discussion, is the problems of connection management, wherein the BipRyt algorithm acts as a decision maker for the distribution of the message load and incoming transmission from external systems. In the next section, we start by describing the classical model of connection management solutions and then demonstrate how the BipRyt algorithm can be deployed to enhance the load distribution within the system. The application of the BipRyt algorithm, is aimed at the problem of the Telecoms, particularly distributed messaging, which differ from the classical use of large scale distributed computing in grand challenges applications. In the domain of messaging, the distributed systems receive incoming packets of (SMS or MMS) of a relatively small size, but at an extremely high frequency. As a result, the management of external connections of a given messaging system, load balancing strategies and routing solutions require new approaches, which are different from the classical load balancing and routing strategies commonly used in distributed computing.

5.4.1 Connection Managers in Distributed Messaging Systems

Connection managers essentially form part of the Communication class of the problem domain that reside at the edge of the system (chapter 2). Their responsibilities are to handle the connections, conversation protocols, security, authentication etc. of the messaging gateway (*see Figure 84*), and they address mainly the communication model between external and internal software entities. The actual transport mechanism of the connection will address the techniques of multiplexing / de-multiplexing and load sharing / balancing across distributed entities.

Connectivity is tightly linked to the mechanism of load balancing which provide a single access point for external entities to access data that might be spread over several physical nodes. A typical load balancer will be using diverse algorithm(s) to direct requests to the appropriate nodes of the system. The distribution processing of the load balancers is completed transparently, with external entities having no knowledge on what exists behind the single access point. From a mobile data perspective, connection management encompasses external areas including connectivity to mobile data applications such as ESME (External Service Message Entity) and VASP (Value Added Service Provider) (SMS06) and

external connectivity to network serving elements such as SMS and MMS. The ultimate aim is to ensure that all functional requirements are serviced to meet a customer requested level of quality of service (non-functional requirements).

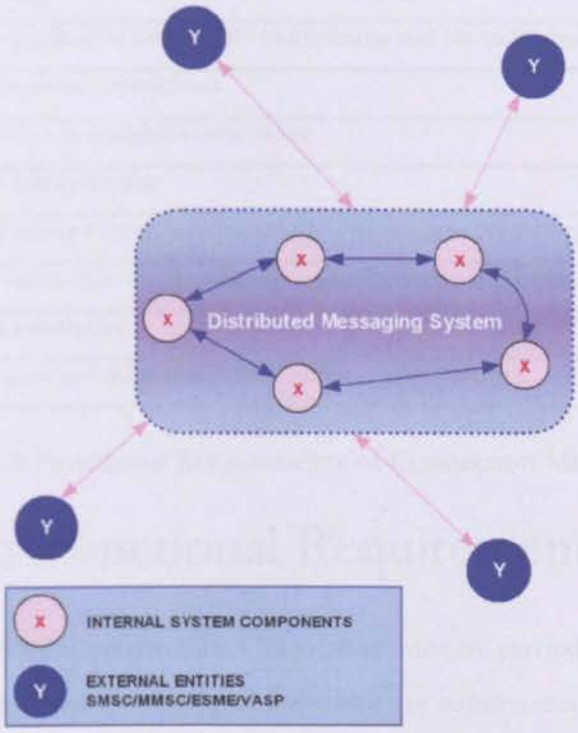

Figure 84 Connectivity within the Telecoms

5.4.1.1 Functional Requirement Model

Table 5 summarises the common functional requirements that the connectivity component of the distributed messaging system, must conform to, in the operability of load balancers within the Telecoms network. In recent years the feature of Protocol Datagram Unit (PDU) load balancing has grown in importance since the number of IP based protocols has increased over the native Telecoms network (NSN07). PDU load balancing means that the load balancing mechanism should be able to gather and manage lower IP packets, until the complete PDU of an upper layer protocol is constructed, that can then be distributed to the nodes.

No	Functional Requirement
1	Send and receive IP packets
2	Perform network level "Per Request Load Balancing"
3	Perform application level "PDU Load Balancing"
4	Perform application level "PDU Multiplexing and De-multiplexing"
5	Manage external connections
7	Share load over available connections
8	Perform authentication
9	Perform atomicity with transparent PDU transaction processing
10	Process connection requests based on user-defined configuration
11	Runtime monitoring
12	Share connection information over nodes

Table 5 Functional Requirements of Connection Managers

5.4.1.2 Non-Functional Requirements – CTQs

The agreed non-functional requirements, CTQs, that must be carried out by the connectivity component of the distributed messaging architecture are summarised in Table 6. The CTQs for connectivity has been established through brainstorming session by looking at the domain knowledge of the Telecoms as well as the state of art (Gard07). These CTQs are commonly addressed within the realm of Telecoms operators and reflects mostly to the Voice of Customer when dealing with the mobile community.

No	CTQs
1	Uniform load sharing
2	Single access point
3	Availability
4	Quality of service
5	Protocol support
6	Design
7	Usability
8	Scalability
9	Reusability
10	Supportability

Table 6 CTQs of Connectivity

5.4.1.3 The House of Quality (HoQ) Analysis

The functional requirements have been mapped to the non-functional requirements into a HoQ matrix (*see Figure 85*). The primary objective is to drill down the high level CTQs, in order to formulate the measurable and controllable quality attributes. As explained earlier, the HoQ is an ideal tool to perform the CTQ flow down process.

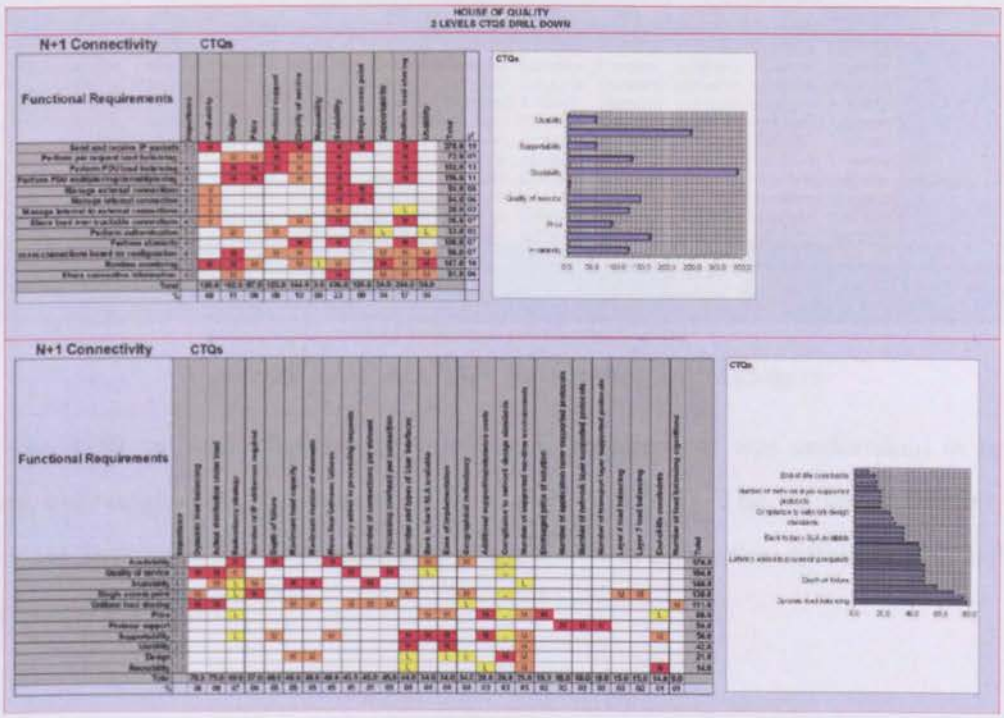

Figure 85 House of Quality for Connection Managers

Level one of the HoQ matrix shows the impact that the CTQs (in the x axis) have over the functional requirements (y axis). Using the feature of CTQ drill down, the CTQs in level one (the Xs) become the (Ys) of the level two. At this level each newly discovered CTQs are atomic, meaning they are controllable and measurable by an experiment. The graphs of both matrices show the importance of the CTQs for each functional requirement. The importance is unravelled by using the Analytical Hierarchy Process techniques as shown in Figure 86.

5.4.1.4 Analytical Hierarchy Process Evaluation

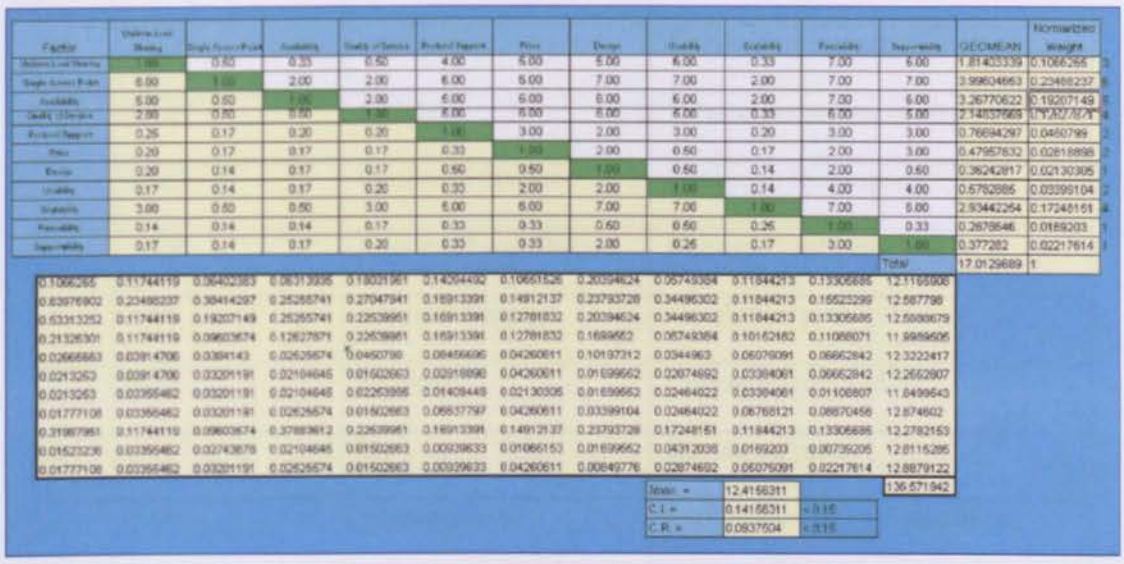

Figure 86 AHP of CTQs for Connection Managers

Using the AHP, an evaluation of the measurable requirement was undertaken in order to measure and weight the importance of the CTQs (Table 7). The values associated to the weights are later used for benchmarking purposes to evaluate the different load balancing strategies and solutions.

No	Importance [%]	CTQ	Unit of Measure
1	23.48	Single Access Point	Maximum number of connections
2	19.20	Availability	Redundancy strategy
3	17.24	Scalability	Maximum number of elements
4	12.62	Quality of Service	Maximum load capacity
5	10.66	Uniform Load Sharing	Dynamic load balancing
6	4.50	Protocol Support	Number of protocols
7	3.39	Usability	User interfaces and API
8	2.81	Price	Envisaged price
9	2.21	Supportability	Back to back SLA available
10	2.13	Design	Design architecture

Table 7 CTQs and Unit of Measures for Connection Managers

5.4.1.5 Load balancing Strategies

In Figure 87, we modelled and illustrated the approach to integrate load balancing designs into the connection manager component of the messaging system. Commonly, the sessions initiated into and out of the messaging system are regulated by either the Load Balancer or connection manager components.

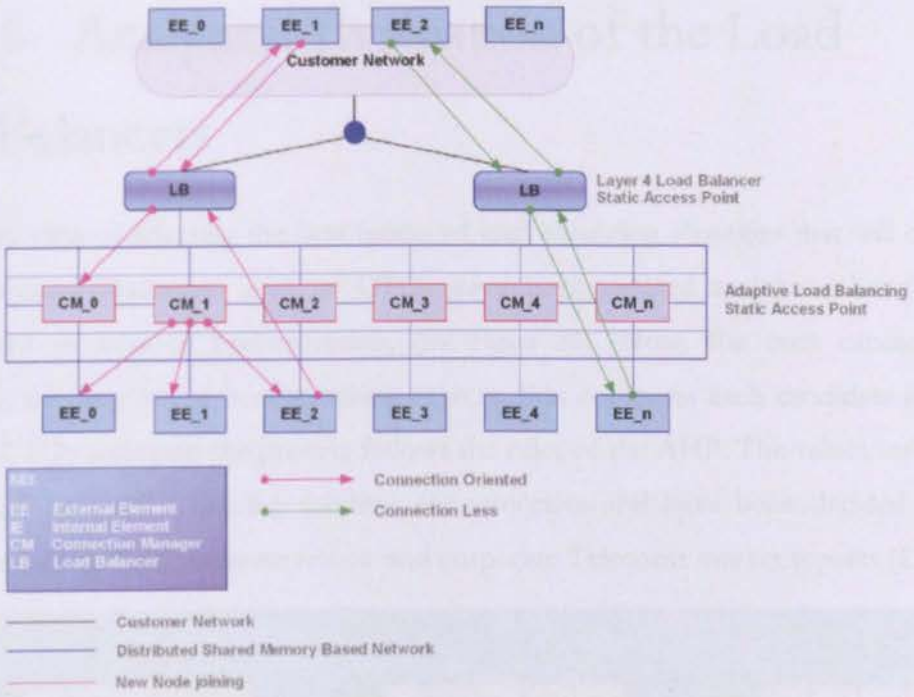

Figure 87 Connection Management Model

In order to design a robust solution for the management of connections to the system, the following four load balancing solutions were selected for further analysis and the rationale behind their choices is because these load balancer best conform to the CTQs proposed and analysed in Table 7, since they are the most commonly employed in Telecoms networks (Zeu04, B-IP06, Cis06a).

Name	Details
Zeus Load Balancer	Advanced software load balancer that runs on multiple platforms.
Linux IPVS	Free code base offering basic load balancing functionality.
F5 BIG-IP LBL	Hardware/software load balancer used by many big organisations around the world.
CISCO CSS11501	The most common networking manufacturer. This hardware load balancer is used by many big organisations around the world.

Table 8 Load Balancing Strategies in the Telecom Networks

5.4.1.6 Analytical Evaluation of the Load Balancers

In the firm view of selecting the best model of load balancing strategies that will connect to the Connection Managers, each of CTQs have been defined and have been weighted through the process of brainstorming, (*see Figure 86*). Now, for each candidate (load balancers), we carry out at benchmarking exercise that compares each candidate against the weighted CTQs and again the process follows the rules of the AHP. The values associated to each CTQS per load balancing solution are subjective and have been decided based on domain knowledge and literature review and corporate Telecoms market reports (Del07).

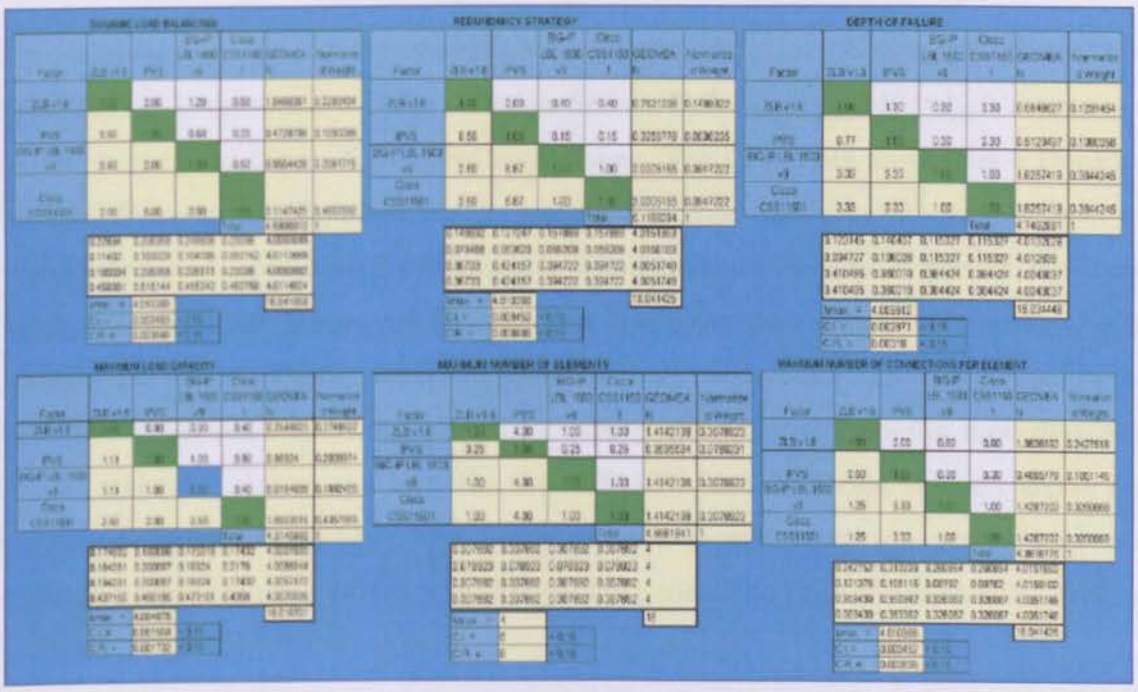

Figure 88 AHP Benchmark of Load Balancers CTQs

Observations

Using the AHP, each measurable requirement criteria was analysed against all of the load balancers, as highlighted in Figure 89.

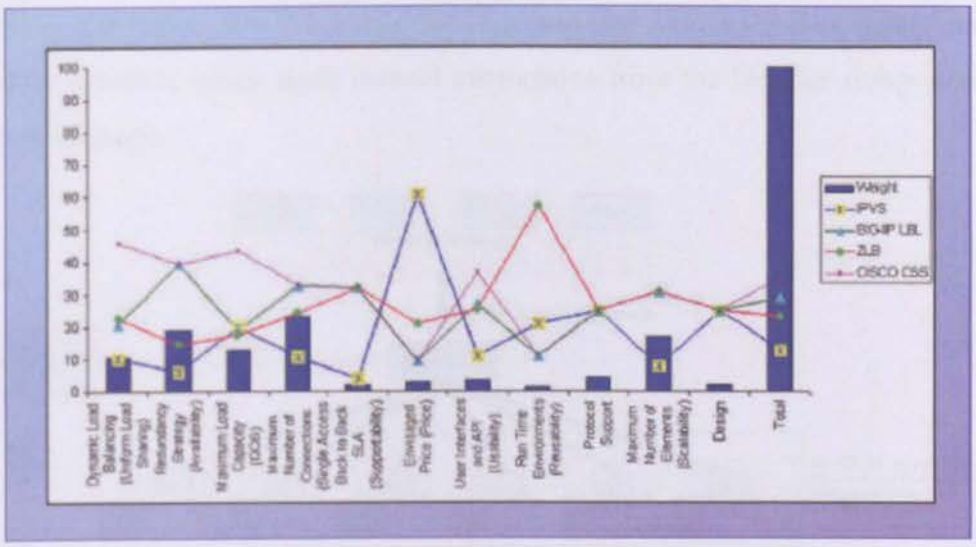

Figure 89 Observations of AHP Benchmark for Load Balancing CTQs

On the grounds of the AHP evaluation, we observed that Cisco CSS series achieved the highest score. In particular, it offers Adaptive Session Redundancy (ASR), which maintains a session even in the result of an active node failure. The CSS series also offers a Dynamic Feedback Protocol (DHP), which enables true dynamic load balancing with servers deployed on back-end nodes providing load balancing information to be used as part of the actual distribution algorithm. Having selected the best load balancing candidate, we now move to comparing it with the BipRyt algorithm using the three most common strategies used in the Telecoms which are round robin, least connections and adaptive response time strategies. This is achievable due to the fact that the CSS11501 load balancing solution allows the network designer to configure the Local server selection based on server load and application response time, as well as traditional least connection and round-robin algorithms (Cis06b).

5.4.2 Application of the BipRyt Algorithm within the Connection Managers

The BipRyt algorithm acts as a load balancing strategy within the Connection Manager where it distributes the load of the messages from external transmissions to the available nodes. Thus, it has been integrated into the model of the connection manager (*see Figure 87*) to enable the decision making process of the algorithm within the operations of external

connectivity (*see Figure 90*). As it can be observed the load balancers (LBs) have been replaced by a switch, which reads control instructions from the Decider component of the BipRyt architecture.

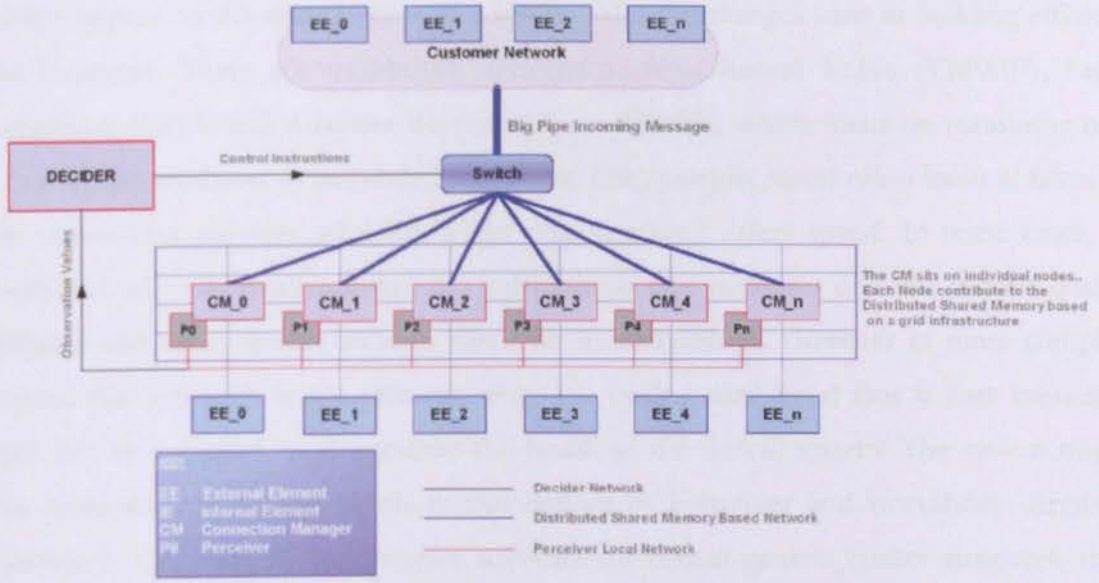

Figure 90 The Integration of BipRyt within Connectivity

The functioning of the algorithm is driven by the assignment of a set of quality attributes (human fabricated AHP) by the system designer, which are founded on the Voice of the Customer and recorded by the Decider. The Perceivers, (P0 ... Pn), are instructed to read the actual values (energy levels) of each quality attributes for each node and send the list of the values to the Decider. The latter uses the BipRyt algorithm to devise a distribution matrix that defines the control instructions for the switch.

In order to evaluate the BipRyt algorithm and compare it against other strategies, we devise an experiment to specifically address the connectivity aspects between external and internal entities. In view of successfully tackling connectivity in a distributed environment we are looking at primarily enabling **horizontal scalability** of the nodes and the ability to multiplex mobile data applications over a number of servicing nodes i.e. **service availability**. We are also looking at **resource efficiency** by optimising asset utilisation, in simple words to get the most out of the messaging infrastructure. Load balancing can help improve system scalability by ensuring that client application requests are distributed and processed equitably across a group of servers. Likewise, it can help improve system availability and resource efficiency by

adapting dynamically to system configuration changes that arise from hardware or software failures.

In parallel and distributed computing, the approaches accountable for data distribution have a direct impact on the effectiveness of a system. Many challenges exist in building efficient load balancers. There are established strategies such as Round Robin (VMW07), Least Connection (Cis07) and Adaptive Response Time (Wes95), which focus on measuring one or two quality attributes of the underlying nodes. For example, round robin looks at fairness; least connection provides reliability whilst response time offers speed. In some cases, in simple and relatively small systems, it is sufficient to take measures of some generic quality attributes and make simple decision based on those readings. However in more complex systems, this approach is not efficient, since the quality attribute(s) that is /are measured might not be sufficient to characterise the health of the overall system. The system might have more than one quality attribute that defines its behaviour and workability. Another limitation is that since these strategies advocate for certain generic quality attributes, they may have an adverse impact on other quality attributes which might be of utmost importance given certain situations. Consequently, in the BipRyt approach, more than one quality attribute is questioned and assessed to maximise the effectiveness of work load distribution. The following experiment compares the BipRyt with the Cisco CSS11501 configured to run on 1) the popular Round Robin algorithm, 2) the least connections algorithm and 3) the adaptive response time algorithm.

5.4.2.1 Load balancing Strategy – Round Robin

Round Robin distribution is one of the most popular algorithms due to its simplicity and efficiency. It is an even request distribution algorithm and the basic principle behind it is to distribute the request forwards (not the load) evenly.

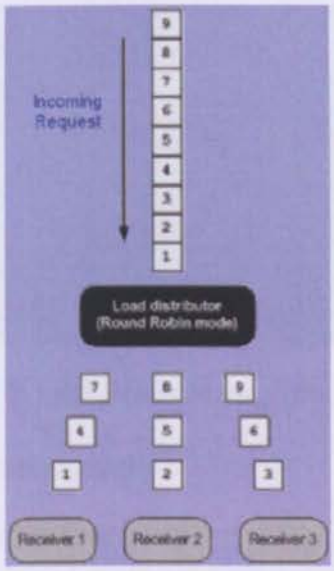

Figure 91 Round Robin Model

The operational model of the round robin, its advantages and its simplicity can be clearly seen in Figure 91. There is no processing overhead involved; every next packet is forwarded to the next receiver until the last is reached, then the forwarding starts again from the first receiver. In most of the cases, where the receivers are equally capable and the loads are low to mid intensity with few load peaks, this algorithm performs very well (Teo01). However it is not resistant to the following:

Constantly high load

When the load stays in low to mid range, any differences in receivers' capacity do not get emphasized, because eventually receivers have enough time to process all the requests and it can also survive short bursts of high load requests. However the algorithm does not perform very well when the system load stays at the high mark for a long time. Usually the receivers are not equally capable so when the load is high and all receivers are operating at their limits, all the differences in capacities become critical. This is illustrated in Figure 92 where "Receiver 2" keeps up with the expectations when running with the moderate load, but fails when the load is increased.

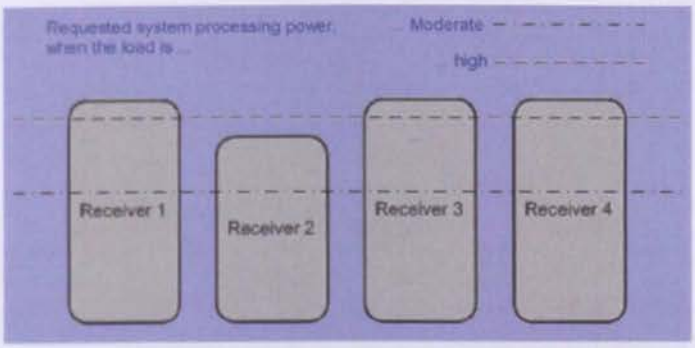

Figure 92 Round Robin Management of Load

Differences in receivers' capabilities are high

The round robin algorithm is also very sensitive to the differences between processing receivers (*see Figure 93*). The higher the differences are, the lower the overall system capacity is, since the maximum load acceptable by each receiver equals to the maximum processing power of the least capable receiver.

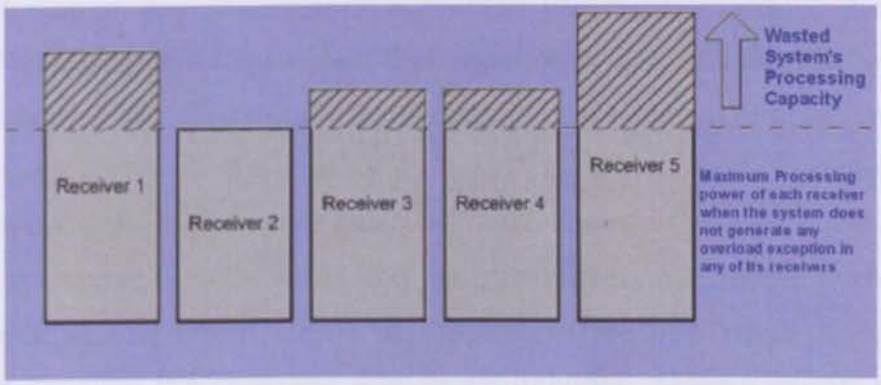

Figure 93 Differences in Receivers Capabilities

Adjustment to load

Another problem, imposed by the inability of the algorithm to adjust the load accordingly, is so called "*hot spots*" in the cluster, where one or more receivers start running at or near their maximum processing capacity. This situation is especially dangerous in distributed systems where receivers need to communicate a lot amongst themselves. Appearance of the "*hot spots*" might cause a complete system to stale, because most of the more capable receivers will find themselves waiting for a response from the overloaded ones, as illustrated in Figure 94 where receivers 1, 3, 4 and 5 might find themselves waiting for Receiver 2 and 6 to respond, (*see Figure 94*).

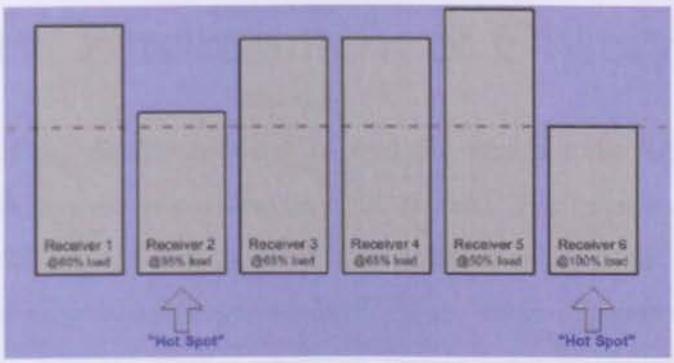

Figure 94 Appearance of "hot spots" in Round Robin

5.4.2.2 Load balancing strategy – Least Connection

Least Connection algorithm maintains a list of active connections (or requests) at any given time to each receiver. Any sub-sequential requests are being sent to a receiver that has a least amount of active requests/connections. This algorithm is quite efficient when requests are equal in "weight". This means that all requests take the same amount of processing power to cope with them. As long as the requests are "equal", Least Connection algorithm performs well in distributing the load, because number of active requests (or pending requests) shows how busy the system is. This works very well for systems that receive loads of similar requests, such as simple web servers. If the requests are not "equal" in weight Least Connection algorithm fails to achieve even load distribution. "uneven weight" requests are request that take different amount of processing power to process them.

5.4.2.3 Load balancing strategy – Response Time

Response Time algorithm uses one of the systems parameter to determine the system load. This algorithm assumes that Response Time is directly related to system load, which is true in most of the cases of simple systems such a server running a web application, but would fail for more complex system, such as multi-tier systems with front-end/back-end architecture, such as messaging systems. If the load balancer uses Response Time to measure system performance it might get a relatively quick response from the front end components, whilst the back end components are actually busy processing previous requests.

5.4.3 Test Environment & Observations

The primary aim of the experiments was to find out whether the BipRyt algorithm is a "*better*" decision making mechanism than its counter parts. The experiment is structured by order of 1) testing the sensitivity of the BipRyt algorithm to adapt to the AHP trend (human fabricated AHP), 2) to compare the adaptability of the other algorithms against the AHP trend, and 3) to test how the BipRyt adapts to system capability against the other 3 load balancing strategies.

Experiment 1: BipRyt Adaptability to AHP Trend

Since the BipRyt uses an AHP model, that represents the user's quality expectations, to control the behaviour of an observed system, the aim of the first test is to verify if the BipRyt follows the trend of the AHP configuration. This is defined by the user who assigns priorities to the CTQs which for this experiment are *Q1*, *Q2* and *Q3*. An example of quality attributes is 1) *Q1 = no. of message per second*, 2) *Q2 = response time* or 3) *Q3 = CPU load*, running on *11 nodes*, each of which keeps a quality container per quality attribute.

The aim of the experiment is to verify whether the BipRyt algorithm responds to the AHP configurations (settings), when making decision for a system. The assumptions made were that 1) the human fabricated AHP has three quality attributes which are number of messages per second, the response time of a node per request and the CPU processing load per message received; 2) the overall system has 11 nodes; 3) each server has three quality containers for each quality attribute above mentioned; 4) each container have three thresholds which correspond to the HRA, MRA, LRA (*see section 5.3.1*) and these thresholds partition the container into 3 equal sized risk areas. There is an indicator mark that fluctuates across the areas illustrating how much energy a particular node has to spend for the specific quality. This means that the more hits to the high risk area for a given CTQ, the less capable the system is to preserve the concerned quality attribute. The mark is set in the low risk area at the initialisation phase. The system is able to read sms input from a message gateway, which is the source of message generation. We started the experiment by setting up the quality priority of the human fabricated AHP to Q1 = 0.6, Q2 = 0.2, Q3 = 0.2. The system is then exercised for this given configuration, and we refer to this as configuration 1, (*see Figure 95*).

Factor	Q1	Q2	Q3	GEOMEAN	Normarized Weight
Q1	1.00	3.00	3.00	2.08008382	0.6
Q2	0.33	1.00	1.00	0.69336127	0.2
Q3	0.33	1.00	1.00	0.69336127	0.2
			Total	3.46680637	1

0.6	0.6	0.6	3
0.2	0.2	0.2	3
0.2	0.2	0.2	3
λmax =	3		9
C.I. =	0	< 0.15	
C.R. =	0	< 0.15	

Figure 95 AHP Configuration 1

After a run of 175650 messages received by the system, the quality priority of the human fabricated AHP was changed to a new configuration, configuration 2. The changed values were Q1 = 0.2, Q2 = 0.6, Q3 = 0.2, and the system was exercised again for another run of 175650 incoming messages.

Factor	Q1	Q2	Q3	GEOMEAN	Normarized Weight
Q1	1.00	0.33	1.00	0.69336127	0.2
Q2	3.00	1.00	3.00	2.08008382	0.6
Q3	1.00	0.33	1.00	0.69336127	0.2
			Total	3.46680637	1

0.2	0.2	0.2	3
0.6	0.6	0.6	3
0.2	0.2	0.2	3
λmax =	3		9
C.I. =	0	< 0.15	
C.R. =	0	< 0.15	

Figure 96 AHP Configuration 2

As the reader can see, the test was dichotomised into two configuration setups, and for each of them, we examine the distribution of the mark of each CTQ of each server across the 3 areas of risks.

We observed that for the test for configuration 1, where Q1 is of highest priority, the number of HRA hits for Q1 is kept to 161 hits over 175650 messages resulting to a percentage yield of 0.091%, which is very small. Furthermore, when the BipRyt algorithm was exercised for the configuration 2, where Q2 is of highest priority, we observe that the number of HRA hits for Q2 is kept to 182 hits over 175650 messaged resulting to a percentage yield of 0.10%, (*see Figure 97*).

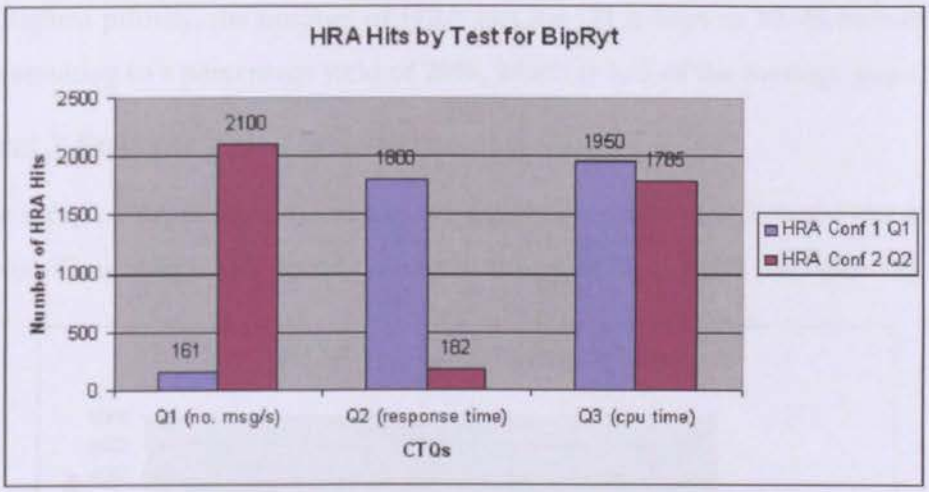

Figure 97 BipRyt Adaptability to AHP CTQ Priority

Experiment 2: Round Robin Adaptability to AHP Trend

The same setup of experiment 1 was used to carry out experiment 2 to test the adaptability of Round Robin to the AHP trend. However, we only used the setup of configuration 1, i.e. Q1 = 1, Q2 = 0, Q3 = 0 and the same of amount of messages, 175650, were injected to the system, (*see Figure 98*).

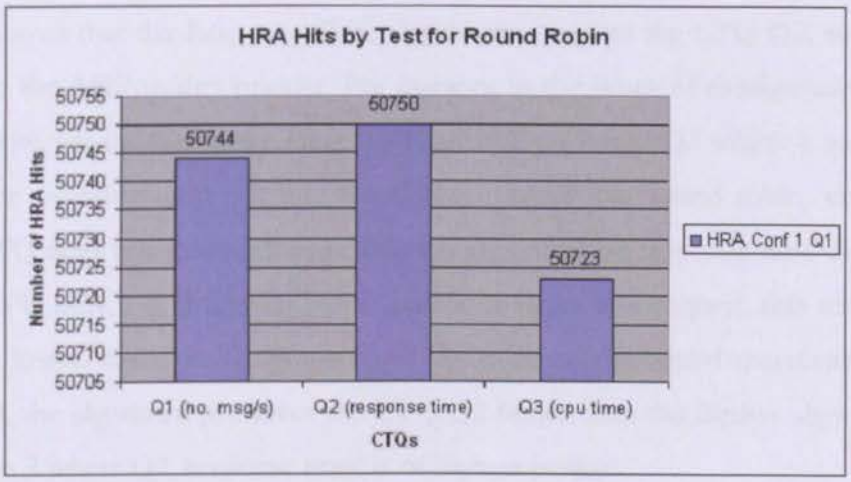

Figure 98 Round Robin Adaptability to AHP Trend

The graph shows that the number of hits to the HRA has considerably increased with Round Robin. The algorithm shows no awareness of the AHP trend and as a result the distribution of the CTQs hits is ad hoc. We observed that the test for configuration 1, where

Q1 is of highest priority, the number of HRA hits for Q1 is kept to 50744 hits over 175650 messages resulting to a percentage yield of 29%, which is 1/3 of the message population.

Experiment 3: Response Time Adaptability to AHP Trend

The same setup of experiment 1 was used to carry out experiment 3 to test the adaptability of Response Time to the AHP trend on both configurations 1 and 2, (*see Figure 99*).

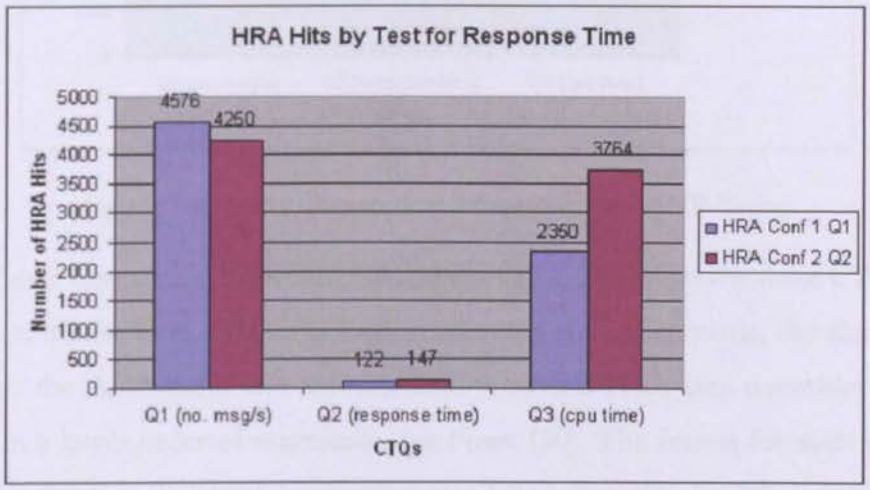

Figure 99 Response Time Adaptability to AHP Trend

The graph shows that the Response Time algorithm preserves the CTQ Q2, response time, regardless to the AHP quality priority. For instance in the setup of configuration 1, Q1 has highest priority, yet the Response Time algorithm still preserves Q2 which is response time. However, the overall number of hits is still much lower that round robin, since response time as a CTQ does positively influence other CTQs, because response time does indirectly reflect on CPU time, e.g. if a given server responds faster to a request, this means that the CPU load is lower. Since the Response Time algorithm was designed specifically to operate on the CTQ, the algorithm preserves the CTQ Q2 better than the BipRyt algorithm for the configuration 2 where Q2, response time, is of highest priority.

Experiment 4: Least Connection Adaptability to AHP Trend

The same setup of experiment 1 was used to carry out experiment 4 to test the adaptability of Least Connection to the AHP trend on both configurations 1 and 2.

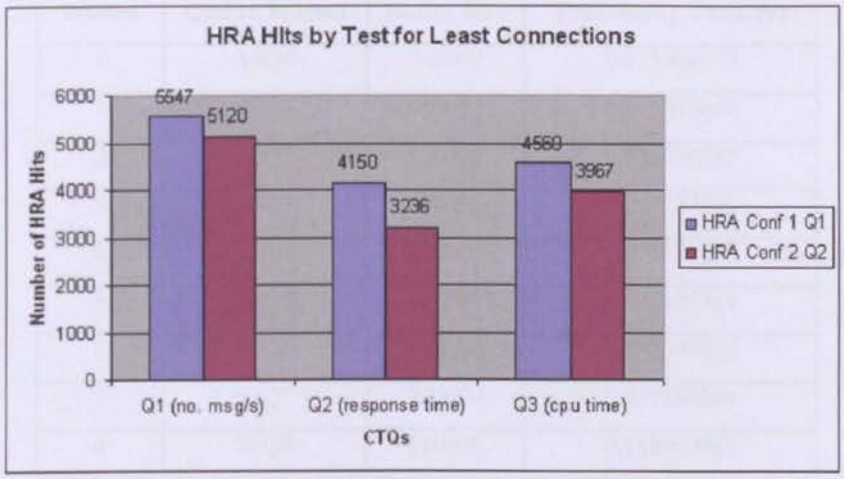

Figure 100 Least Connection Adaptability to AHP Trend

As for the Least Connection algorithm, since it does not depend on the three CTQs, Q1: no. msg/s, Q2: response time, Q3: CPU load, chosen for the experiments, the algorithm does not adapt to the AHP trend and the distribution of the HRA hits resembles the Round Robin, but in a lower order of magnitude, (*see Figure 100*). The reason for such lower hits is again, due to the fact that least connection as a CTQ may also positively influence other CTQs such as CPU time and response time, but not always. For example, when a given switch distribute the load based on least connections, the next requests are forwarded to the server with the least amount of connections, hence less amount of message is being transported and processed on that server, preserving the CTQ CPU time. As a result, the number of HRA hits for CPU time is reduced.

Experiment 5: BipRyt Adaptability to the Capacity of Server

We carried out an experiment to check how the BipRyt algorithm behaves when connected to nodes with different capabilities in terms of 1) processing power, 2) memory capacity and 3) no. of read/write to a database. Since the nodes on our test bed were homogeneous, in order to simulate the difference in node capabilities, we designed 11 queues, on each node and each queue had different configurations as shown in Table 9. This will allow us to demonstrate the adaptability of the different load balancing algorithms in a real life heterogeneous environment.

Nodes	Queue System	Buffer Size	Processing Time (s)
0	S0Q0	940000	0.015904573
1	S1Q1	860000	0.017892644
2	S2Q2	780000	0.021868787
3	S3Q3	600000	0.043737575
4	S4Q4	520000	0.071570577
5	S5Q5	470000	0.093439364
6	S6Q6	360000	0.103379722
7	S7Q7	220000	0.119284294
8	S8Q8	110000	0.155069583
9	S9Q9	90000	0.170974155
10	S10Q10	80000	0.186878728

Table 9 Server Configurations

Each queue has different buffer size which represents the memory capacity of the node. We incorporated a processing unit within the logic of each queue which is triggered when a message is loaded into the buffer. We also simulated database I/O by defining write requests to a mySQL database when a message is being processed. So in the experiment we have 3 CTQs which are 1) Q1 = CPU Time, 2) Q2 = memory usage and 3) Q3 = no. of DB I/O. We created an AHP with the setting of Q1 = 0.3, Q2 = 0.6 and Q3 = 0.1 and provisioned the BipRyt algorithm. We deployed the queue applications on 11 nodes and connected each of them to a switch that was linked to a computer (the switch node) running the load balancing algorithms which were 1) BipRyt algorithm, 2) Round Robin algorithm 3) Response Time algorithm, and 4) Least Connections algorithm.

The objective of the experiment was to check the overall adaptability of the aforementioned load balancing algorithms and analyze how each of them distributes the load given heterogeneous server capabilities. The switch node is able to read a message file which is the source of the message generation. Firstly, the system is run whilst the maximum buffer capacity of individual queues per nodes is adjusted until there is no hit to the high risk area of the quality container, given a fix number of message per second. Next, we performed the calibration of the system overall capacity by adding the individual load capacity together.

Having calibrated the server capacities, the system is run for 10 minutes starting with the BipRyt algorithm, then Round Robin, Response Time and finishing with Least Connections.

At the end of a run, the number of hits to the high risk areas for the CTQ, buffer population which defines the memory available or message load for each server is recorded.

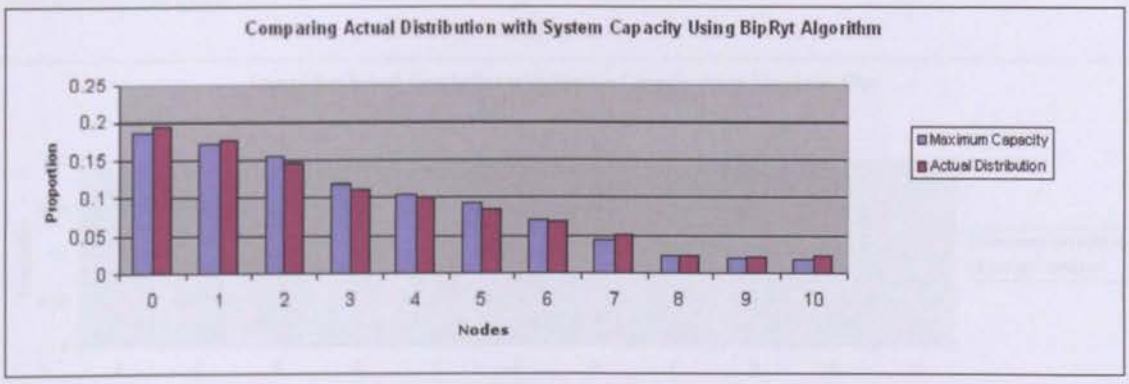

Figure 101 BipRyt Distribution vs. System Capability

When the system was exercised with the BipRyt algorithm, in 10 minutes, the switch distributed 464038 messages to the servers. As it can be observed in Figure 101, the BipRyt algorithm intelligently balances the incoming load as per maximum capability of the nodes. The actual distribution follows the trend of the node capacity, implying that the system is efficient with negligible number of server message starvation and little packet loss. Packet loss is reached when the number of message sent to a node is greater than the buffer size of a given queue.

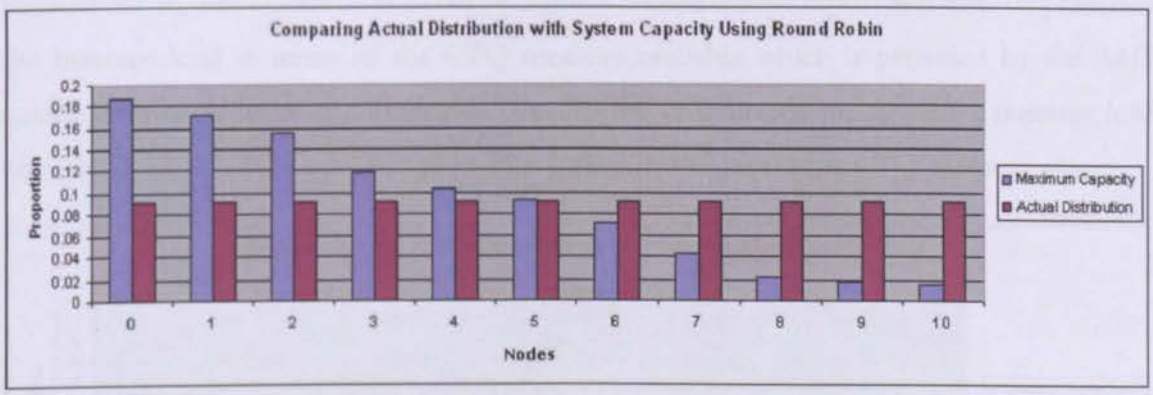

Figure 102 Round Robin Distribution vs. System Capability

As Figure 102 illustrates, due to its simplistic nature, the Round Robin algorithm distributes the load uniformly regardless to the CTQs and the maximum capabilities of the nodes. So at some point in time, Round Robin will starve some the highest capable nodes and overload the weakest. In such an environment, wherein nodes have different load capacities, Round

Robin is very inefficient especially for packet loss. In 10 minutes, with Round Robin, the switch distributed 1698771 messages but many of the packets where rejected by the low performing servers.

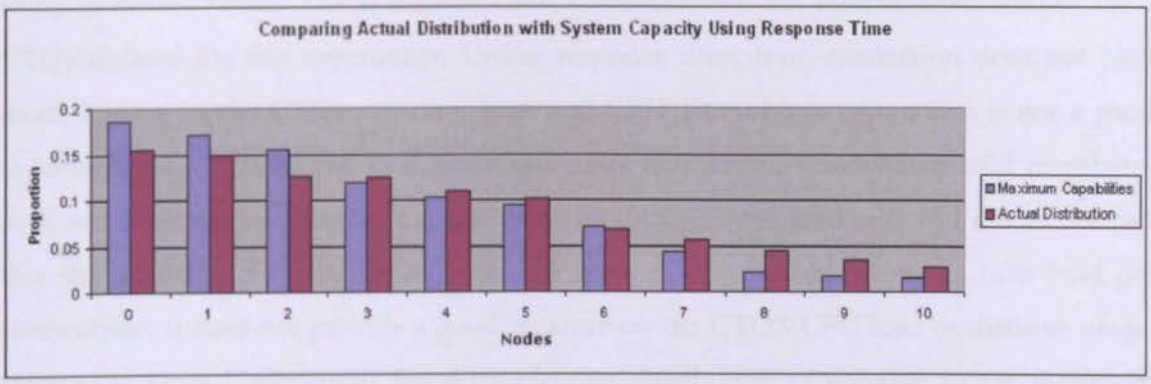

Figure 103 Response Time Distribution vs. System Capability

When the system was exercised with the Response Time algorithm, in 10 minutes, the switch distributed 1494335 messages to the nodes. The graph in Figure 103, shows that the Response Time algorithm does attempt to follow the trend of the nodes' maximum capacities. However, the difference between the nodes' maximum capabilities and the actual distribution of Response Time is still larger than the difference of the BipRyt algorithm, meaning that there is still more occurrences of message starvation and packet loss. This is because the BipRyt algorithm is based on a multi-criteria quality model and directly preserves the message load in terms of the CTQ memory available which is provided by the AHP quality priority. Whereas the Response Time algorithm indirectly preserves the message load by managing the CTQ response time which indirectly influences the CTQ memory usage.

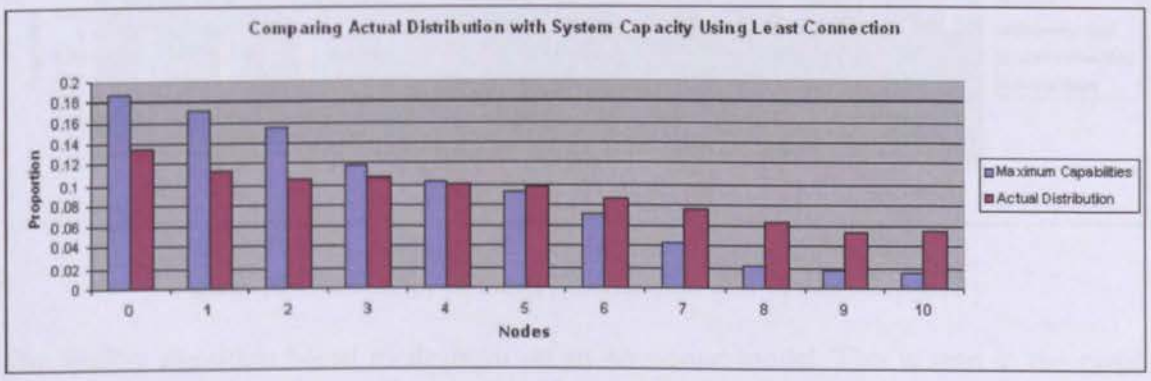

Figure 104 Least Connections Distribution vs. System Capability

When the system was exercised with the Least Connections algorithm, in 10 minutes, the switch distributed 1583827 messages to the nodes. The graph in Figure 104, shows that the trend of the actual distribution for the Least Connections algorithm tends to resemble the trend of Round Robin. This is because Least Connections is not directly influenced by the 3 CTQs defined for this experiment. Unlike response time, least connection does not have much impact on the CTQs memory usage and CPU Time. Least connection is not a good measure for CPU Time and load, since one node may have 5 connections of 2 megabytes each and a second one, having 2 connections of 20 megabytes load each and due to the fact that the algorithm base its decision on the number of connections rather than load per connections, it does not provide a good measure on the CTQS CPU load or memory usage. Hence the Least Connections failed to adapt its distribution of message to the maximum capabilities of the servers, but still provides a better solution than Round Robin.

We finally summarised the efficiency of the four algorithms into a graph, shown in Figure 105. The distributions below the 0 mark of the Y axis show the number of time a particular algorithm is depriving a performing node from messages, whereas the distributions above the 0 mark, show number of time a low performing node is being flooded with messages and these message will be rejected. Packet rejection in the Telecoms is very costly and also regulated by government policies, which means that connection models should not allow for such situation to happen.

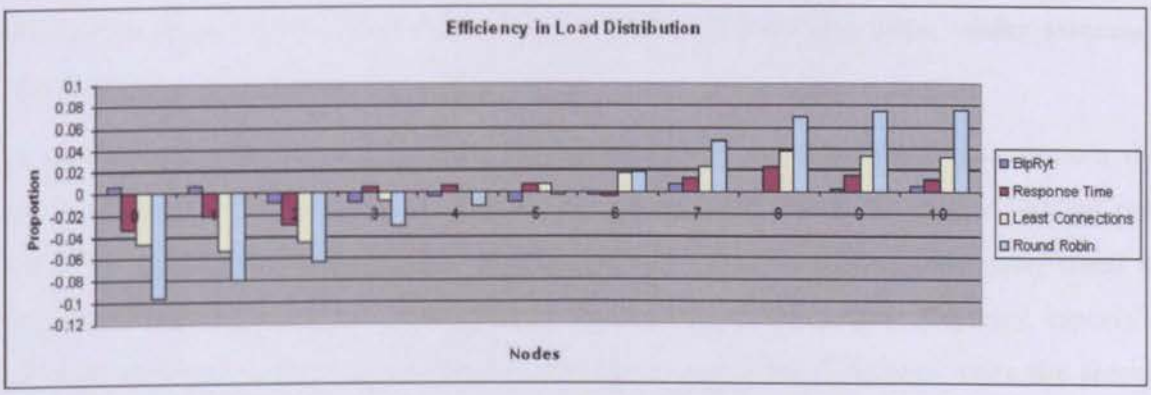

Figure 105 Efficiency in Load Distribution and System Capability

The BipRyt algorithm based its decision on an economic model. This is seen in the graph, since the distribution of message is very rarely rejected by the nodes, i.e. the supply of the message is coordinated by the purchasing power (capability) of the consumers (nodes). As a

result, the algorithm handled fewer messages than the others, since the amount sent is roughly the number of messages that the overall system can accommodate. The BipRyt algorithm, hence, also provides a measure of the system capability which can be used to assess the requirements or needs of the overall system, as its operations evolve. These are not achievable by the Response Time, Least Connections and Round Robin algorithms, since they based their decision on one CTQ parameter solely, instead of a multi-criteria quality model.

5.5 Conclusion

This chapter presented a new multi-criteria decision making algorithm which has been designed for the management, distribution, control and optimisation of systems resources (e.g. nodes, CPU, channel, memory etc). It is an empowerment strategy that provides autonomy to several parts of the systems and its novelty lies in the ability to use a quality model based, not just one or two predefined quality attributes, but on a multitude of quality attributes. The algorithm is based on an automated AHP analysis of the quality model, wherein decisions are taken for the welfare of the system. During the design and implementation of the BipRyt algorithm, we observed that techniques such as the AHP and the HoQ provide instructions that can be automated through a sequence of actions. This is demonstrated by the process of applying the AHP formula into the Decider's code base. Hence, the algorithm has been constructed to represent both Run Time quality assurance and dynamic quality enforcer.

In addition to system related quality criteria, the BipRyt algorithm takes into account the user's perspective of the quality model, i.e. the user can configure the quality features that he/she is expecting from a system. BipRyt ensures the system follows the quality trend of the user, thus preserving the quality model against a potential decline in quality, especially within distributed systems. This is even more significant in the Telecoms, since the service node population and the complexity associated with a message grows in accordance to market demand and evolution. Thus, we demonstrated that the BipRyt algorithm addresses the problem of Lehman's 7th Law of evolution: Decline Quality.

At the implementation phase, within the Perceiver / Decider communication model, the BipRyt algorithm uses a history of data values that represent the energy levels of the CTQs per nodes, thus building up a feedback system based on a quality model, essentially addressing the 8th Law of evolution: Feedback System.

To validate the functionality of the BipRyt algorithm in enforcing a defined quality model, we presented a case study that implements the BipRyt algorithm as a load balancing decision making mechanism within the communication model of distributed messaging systems. We illustrated how classical connection managers are implemented within a messaging business and carried out an experiment to benchmark the most commonly used load balancers within the Telecoms. This started by building a specification chart in terms of functionalities and CTQs of each load balancer. We tested the capabilities of the BipRyt algorithm against the Round Robin, Response Time and Least Connections strategies. The result showed that the BipRyt algorithm efficiently distributed the work load especially in heterogeneous network, with different capability servers. It followed an economic model, balancing the demand for message processing with the capacity of servers (purchasing power).

6 IMPLEMENTATION OF DISTRIBUTED MESSAGING SYSTEMS

And they discovered something very interesting: when it comes to walking, most of the ant's thinking and decision-making is not in its brain at all. It's distributed. It's in its legs.

Kevin Kelly

6.1 Overview

The early 1980s saw tremendous expansion in the area of network deployment (Page01). Yet, by the mid-1980s, certain companies were experiencing growing pains from deploying many different (and sometimes incompatible) network technologies. The problems associated with network expansion affect both day-to-day network operation management and strategic network growth planning. Each new network technology requires its own set of experts, and the staffing requirements alone for managing large, heterogeneous networks created a crisis for many organizations. An urgent need, therefore, arose for automated network management (including what is typically called network capacity planning) integrated across diverse environments.

As a result, there are a large number of technologies that enable heterogeneous interactions between software systems and hardware systems, but most of them address only certain aspects of communication, for instance in messaging infrastructure, these include message sequences and content expressions. Furthermore the meaning of individual messages between systems is normally hardwired into the software used, on an application-by-application basis. It is complicated to design software able to interact effectively with heterogeneous systems that can be dynamically discovered in global scalable messaging network.

In the Telecoms, the value attached to each message entering a message gateway is of vital importance. In such environment, the questions system designers asked are "*How can the meaning of any message be determined?*" "*What is the effect on the real world when sending a particular message such as a request, a confirmation or a delivery?*" In these circumstances, the quality attributes (CTQs) required to service a message are defined and analysed in order to build a system for quality. As we illustrated in chapter 1, within a message based communication, knowledge about the quality attributes can be induced by analysing the significance of a message, the correspondent and the context of a message. Therefore to design and build a distributed system that will transport each and every message to a certain agreed level of service and QoS, we are required to integrate a new model of communication.

In this chapter, we propose a communication model which is inspired from the model of the human conversation (Mak04). The rationale behind the metaphor is the aspect of dynamism within the complexity of human conversation. This, together with the knowledge gained from our findings in previous chapters, resulted into a new approach for development of a distributed messaging system based on an advanced network technology called InfiniBand, exploiting the concept of Remote Direct Memory Access (RDMA) and distributed shared memory.

To explain the implementation of a distributed messaging model within the realm of the Telecoms, we employ a case study that reflects a real challenge to the carriers towards enforcing and managing Service Level Agreements within their systems. This experiment addresses the problem of building a distributed messaging system that will 1) encapsulate the aspect of dynamism; 2) provide a communication model that can easily scale without compromising the manageability of dispersed nodes and 3) integrate the BipRyt algorithm to manage the workload distribution across nodes.

6.2 Communication Model Based on InfiniBand over RDMA Channel Interface

The communication model is at the heart of the proposed distributed messaging System. The model has been built by applying the knowledge gained from the study of the human conversation, providing focus on the dynamic aspect of communication (Mak04). The investigation shows how the communication model has been instantiated from a generic model of the human conversation towards providing the specifications and definitions about the information being transported over the communication medium or the substratum (*see chapter 1*). It also provides the rules governing how the information is accessed (provisioned and retrieved) to and from the communication medium. The ERD in Figure 106 has been derived from the structural design of the human conversation and represents the communication model as an information system. In simple terms, we propose the implementation and deployment of database management system to organise (manage) the structure of the communication model, with the aim of reducing the complexity of managing distributed nodes.

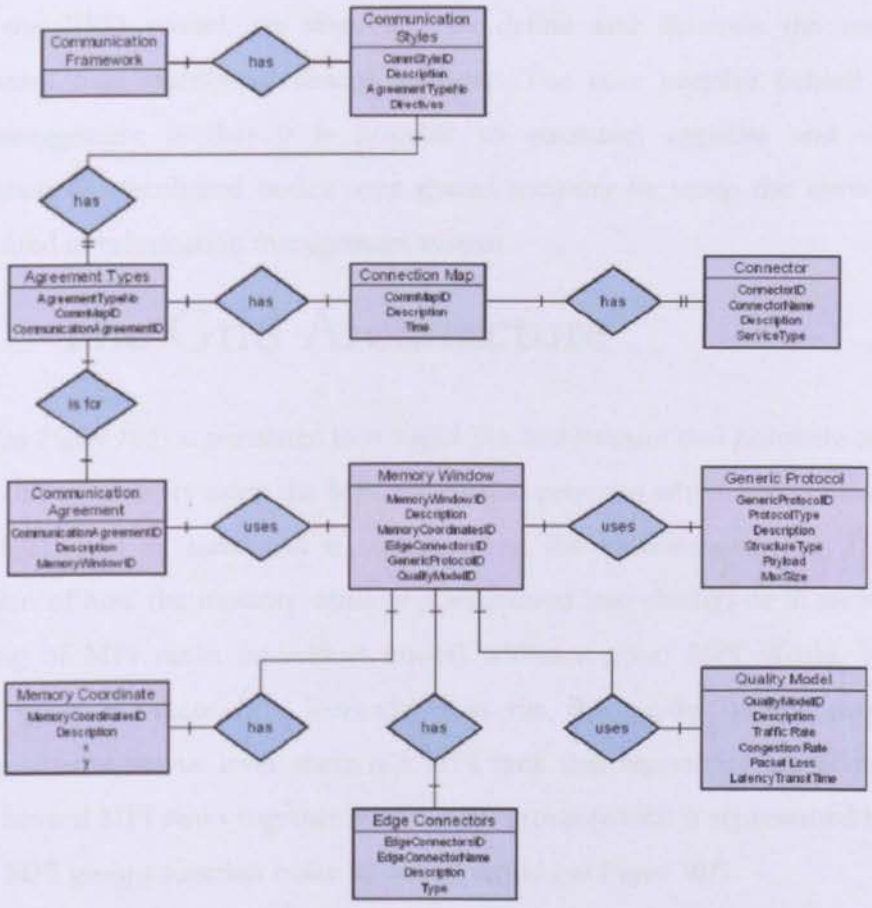

Figure 106 Communication Structural Model Derived from the Human Conversation

As we journey through the ERD, we find that a communication framework is based on different styles (as discussed in chapter 4) which define the directives that govern the communication process. The directives specify the number of participants within the communication, the medium of transport structure (FIFO queues, hash table, array or stacks), the type of message being transferred (syntax and lexical) and finally the quality attributes most significant to ensure high level of service. Upon agreement on the directives, the communication is setup. The connectors are the "conversationalists" over a defined memory window which ensure the deployment of the edge connectors that communicates with external systems. For the purpose of management, the window is uniquely identifiable by coordinates which uses a quality model to establish statistical counters that measure the CTQs as prescribed by the communication style. Finally, to enable heterogeneous communication across different protocol technologies, a generic protocol structure is used to instantiate the medium of transport.

Based on the ERD model, we were able to define and illustrate the mechanics of communication over distributed shared memory. The core premise behind the shared memory management is that it is possible to structure, organise and manage the communication of distributed nodes over shared memory by using the same techniques commonly used in information management system.

6.2.1 The Grid Architecture

The ERD (*see Figure 106*) is translated into a grid like architecture that primarily organises the distributed shared memory using the MPI application protocol where each memory window (*see Figure 108*) is of fix sized and is configured at the initialisation time. The grid is a representation of how the memory window is structured into clusters or in technical terms, the grouping of MPI ranks (individual nodes) within a given MPI World. In the MPI application structure, there is a hierarchy that fits the model of the proposed grid organisation. At the atomic level, there is a MPI rank that represents one address (usually one node). Several MPI ranks together form a MPI group (which is represented by a cluster) and several MPI groups together make up a MPI world (*see Figure 107*).

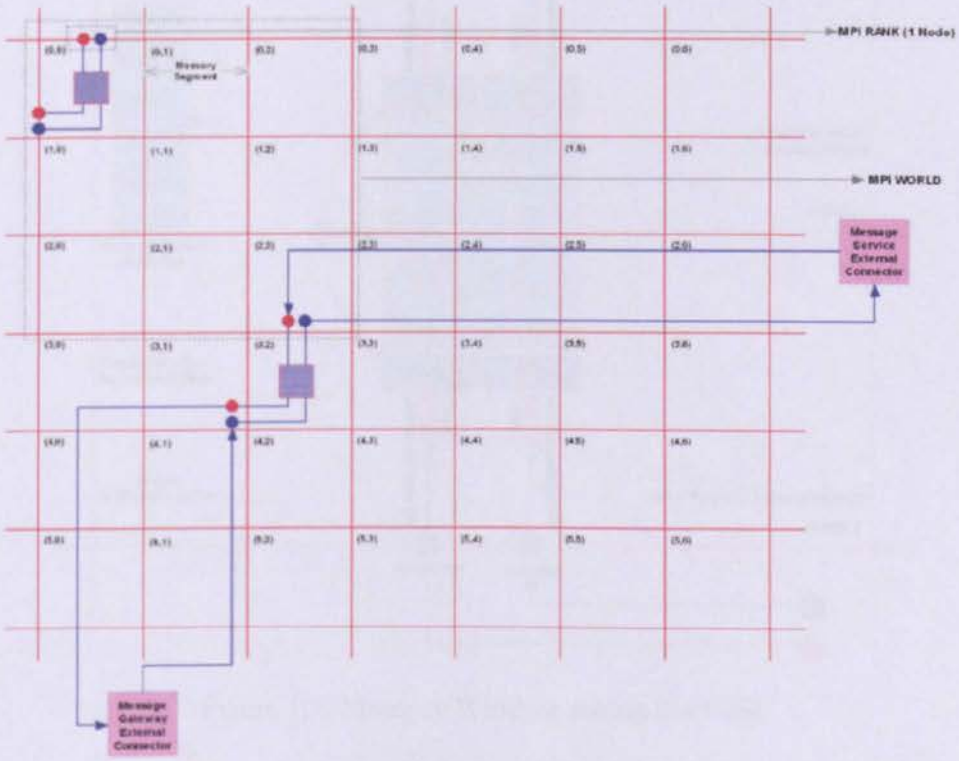

Figure 107 The Grid Representation of the Communication Model

The memory window illustrated above represents the simplest form of communication between two MPI ranks. However, within its structure, it also manages three main features of the conversation, which are quality monitoring, generic protocol management and connector mapping (*see Figure 108*). The quality monitoring system is based on the BipRyt algorithm, which monitors the QoS parameters of inter connected queues, represented by monitor M in Figure 108. A management system is included within the memory window to handle the generic protocol structure which consists of two partitions. The first partition holds information about the message type being transported, e.g. Telecoms protocols would be SMPP, CIMD and UCP or any other user defined protocols. The second partition holds information about the size of the message being transported. The generic protocol structure is written on a buffer that resides on the distributed shared memory, managed by the RDMA Channel Interface and by default, it follows a typical FIFO discipline. The connection mapping is handled by the IB Resource Adaptor edge (*see section 6.3.1*).

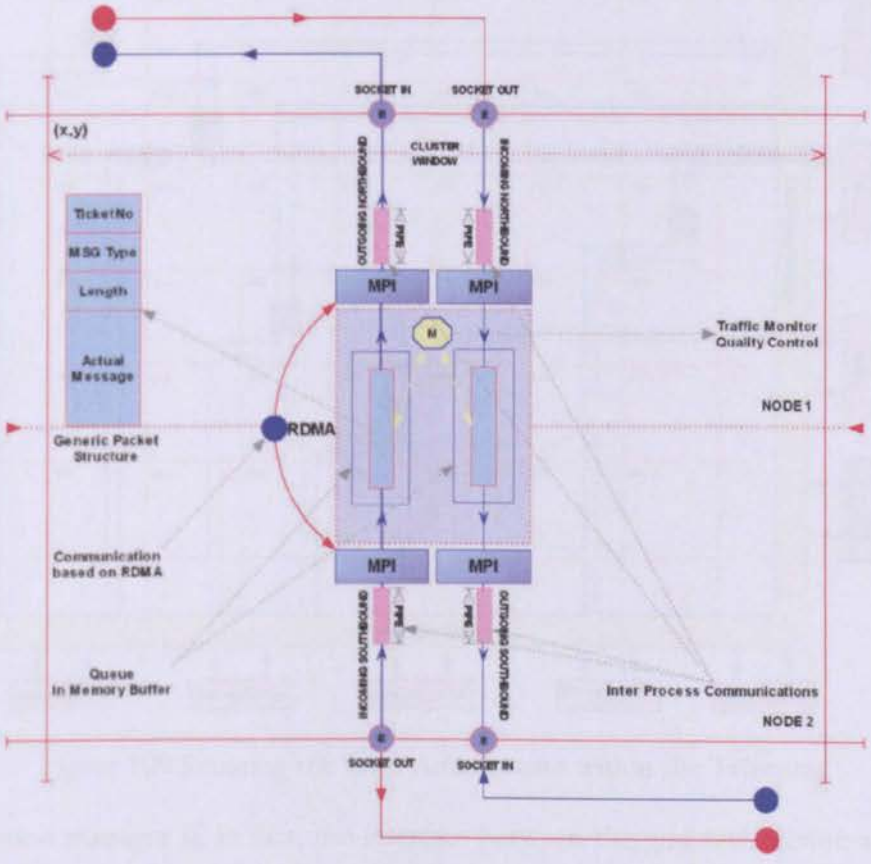

Figure 108 Memory Window within the Grid

6.2.1.1 Integration of the Grid Architecture within Messaging

In this section we integrate the application of connection managers, as discussed in chapter 5, onto the proposed grid architecture. Connection managers form part of the communication class within the problem domain and reside at the edge of the distributed messaging system. We dichotomised the aspect of connection managers into 2 parts. The first is called application connection managers that manage all the external connections originating from an application within the IP based domain. The second part is called network connection managers that manage all the connections that originate from the Telecommunication operators' network (*see Figure 109*).

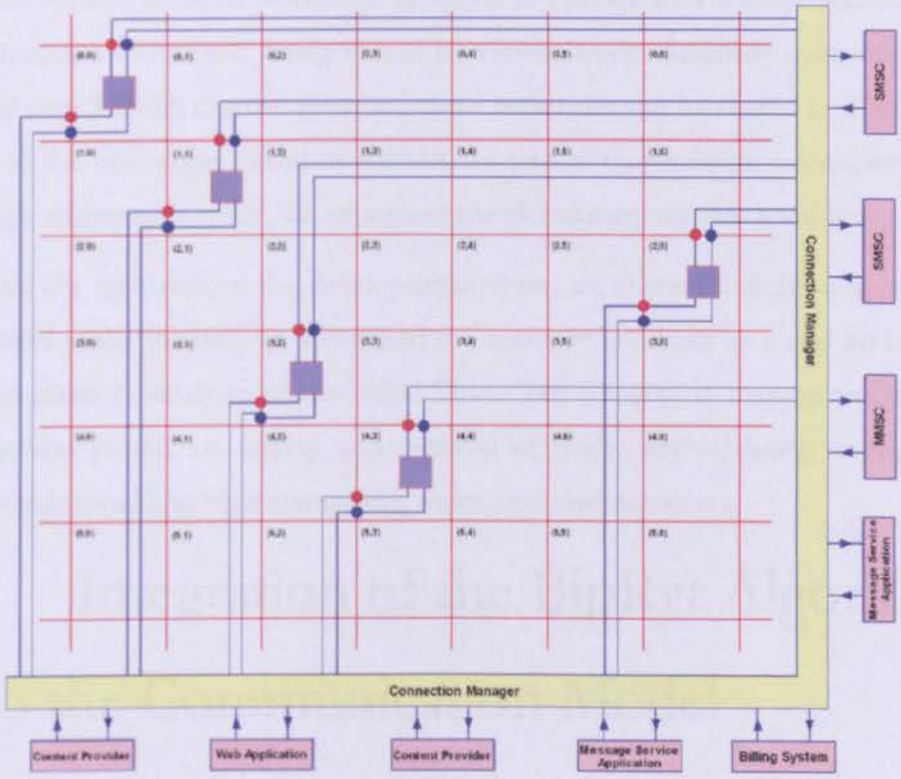

Figure 109 Situating the Grid Architecture within the Telecoms

The connection manager is, in fact, the interface between the grid architecture and external end points such as SMSC and MMSC, content providers and message service applications. The message service application are software that are used to provide services such as message translation, content configuration, generate billing information per message for

billing systems, etc. In summary, the connection managers have the following responsibilities:

- Integration of the BipRyt algorithms for load balancing
- Routing of multipart message to the same destination
- Routing based on connection groups
- Network security layer between applications, the cluster (MPI group), and the network (MPI World)
- Accept protocol agnostic connections from each node in the cluster using the generic protocol structure of the memory window.

As mentioned earlier the management of the communication model can be operated as an information system. In other words it is designed to operate over a single addressable space which constitutes of memory being shared by various nodes (memory contribution). Using InfiniBand over RDMA channel interface, these memories can be shared as if it was a single unit, due to the zero copy model of RDMA for packet transmission (*see chapter 3*). Having built a single addressable space, we structured the global memory into windows.

To illustrate the approach, at the development phase, we created a single addressable space from 11 MPI ranks (nodes); we structured the memory windows as a grid and deployed a very fast in-memory database system called Times Ten (Ong06) to manage and organise the communication model, i.e. adding and removal of nodes and updating attributes of the memory window (adding edge connectors, memory coordinates etc).

6.2.2 Integration of the BipRyt Algorithm to the Communication Model

The BipRyt algorithm has been applied to the communication model by manage the queueing system of each memory window. It is at the heart of the monitor M in Figure 108, as its primary role is to observe the QoS parameters of the traffic for the buffer of the queues. The task of the QoS monitor is to gather the QoS attributes i.e. 1) throughput performance; 2) number of packet rejected; 3) number of packet inside the queue buffer; 4)

the queue's buffer size and 5) the transit time of any given packet within the queue. These are CTQs that we have investigated to check the impact of increasing the throughput and observing its consequences within the buffer population and packet loss (chapter 4). The QoS data are fed into the Decider component of the BipRyt algorithm, which monitors the queue dynamics vis-à-vis a given quality model expressed as an AHP. The BipRyt algorithm will ensure that the queueing mechanism behaves according to the defined quality model. The graphs in Figure 110 demonstrate how the queueing mechanism adapts to an increased performance throughput when the BipRyt algorithm preserves **robustness** of the queue towards avoiding buffer overflow.

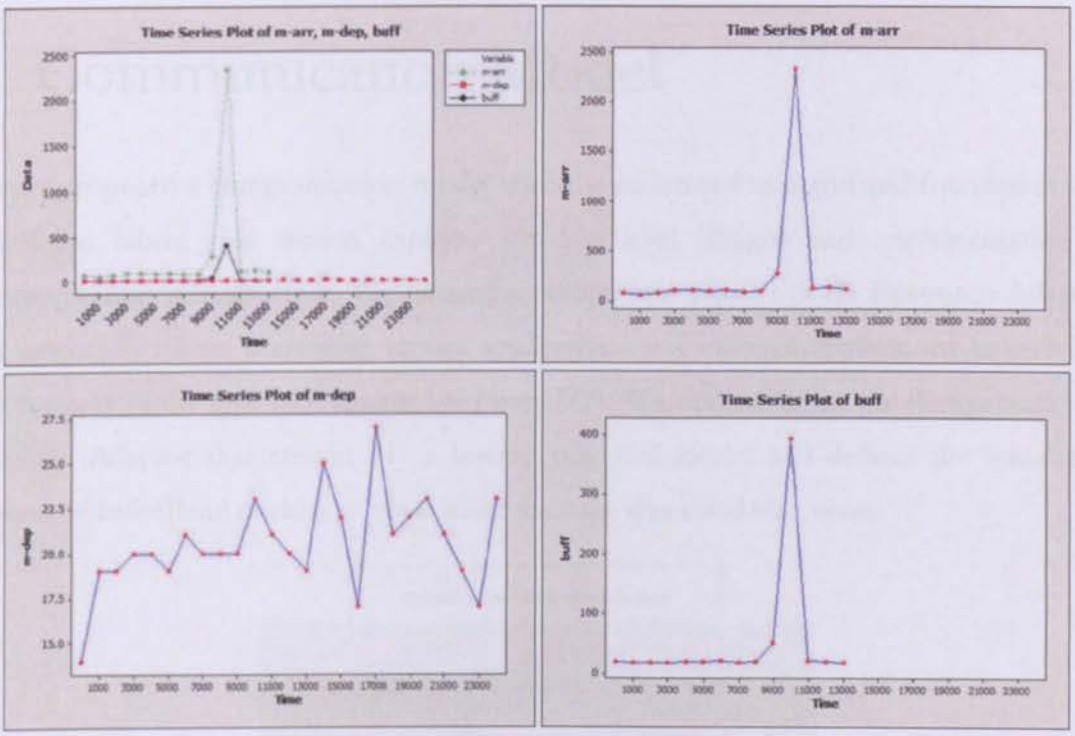

Figure 110 BipRyt Adaptability to different CTQs Configurations

The green line on the first graph shows the arrival rate with a surge in incoming traffic which results in an increase in the buffer population, which is depicted by the black line. The buffer population was configured to accommodate a maximum of 400 packets, and clearly we observe from the graph that the buffer never overflows. This is the robustness policies that the BipRyt algorithm preserves, but consequently it also hurts another aspect of reliability which is packet loss. Since the buffer protects itself, to avoid overflow the queue will reject

any packets during a surge, therefore no matter what the size of the buffer is, it will not overflow.

Yet, in certain circumstances, packet loss, as a CTQ, may be of utmost importance, in which case the BipRyt can be configured to preserve this CTQ by reducing packet loss in dynamically increasing the buffer size of the queues to its natural limit if required. The other 3 graphs are magnifications of the time series of the first graph, i.e. 1) arrival rate, 2) departure rate and 3) buffer population.

6.3 Implementation of the Communication Model

Having proposed a communication model which is structured as a grid and founded on the InfiniBand fabric, this section explains the low level designs and implementation of communication infrastructure. We devised a component called the **IB Resource Adaptor** that essentially allows messaging service applications and external applications to interface and connect to the grid architecture (*see Figure 111*). We document on the design of the IB Resource Adaptor that consist of a layered protocol model and defines the translation process of InfiniBand packets to application message types and vice versa.

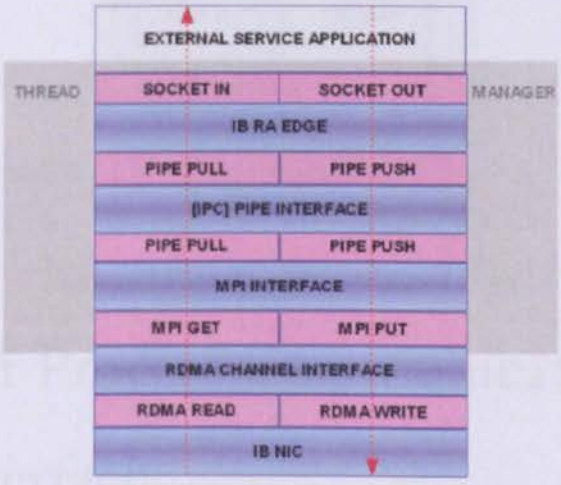

Figure 111 IB Resource Adaptor

The following paragraphs elaborate on each individual layer of the IB Resource Adaptor.

6.3.1 InfiniBand Resource Adaptor Edge (IB RA EDGE)

IB RA Edge is a simple TCP IP client-server application with two connection flows, residing at the edge of the InfiniBand fabric and external application such as Java.

First connection flow:

Messages are pushed from an external source to the socket interface, which in turn is pushed to the IPC Pipes. A socket listens for connection, and read data over the line, which is placed onto a buffer and then transformed into a byte stream. Finally the byte stream is sent over the IPC Named Pipes.

Second connection flow:

Messages are pulled from IPC Pipes by the socket, whilst a listener constantly checks if there is data in the pipe. If there is data, it is then read from the pipe to a buffer which is transformed into an object and sent over the socket.

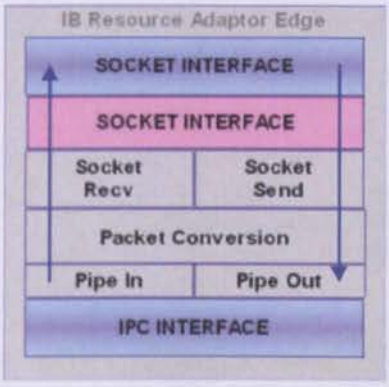

Figure 112 IB Resource Adaptor Edge

6.3.2 Inter Process Communication (IPC) Pipe Interface

IPC Named Pipes are an UNIX inter-process communication mechanism that is a standard part of POSIX.1. Whilst standard pipes can only be used between related processes when a

common ancestor has created the pipe, named pipes allow unrelated processes to exchange data. A named pipes is actually a type of file, with the standard file I/O functions (open, close, read, write, unlink, etc.). Using a named pipe for a .signature file instantly allows applications to access a randomly created .signature, without requiring modifications to the individual applications. On the application perception, the .signature file acts just like a regular text file. When a FIFO is opened, the non-blocking flag (O_NONBLOCK) can be configured.

The main task of the UNIX IPC Named Pipe interface, in our context, is to transport bytes from the IB Resource Adaptor Edge to the InfiniBand network (*see Figure 113*). The IPC pipe interface dynamically allocates memory to a buffer using the number of bytes to be sent, and bytes by bytes fills the buffer from the named pipes. When a pipe is created, one can specify whether the pipe is inbound or outbound which is a configurable parameter within the arguments of the createPipe() function which primarily avoids bytes collision.

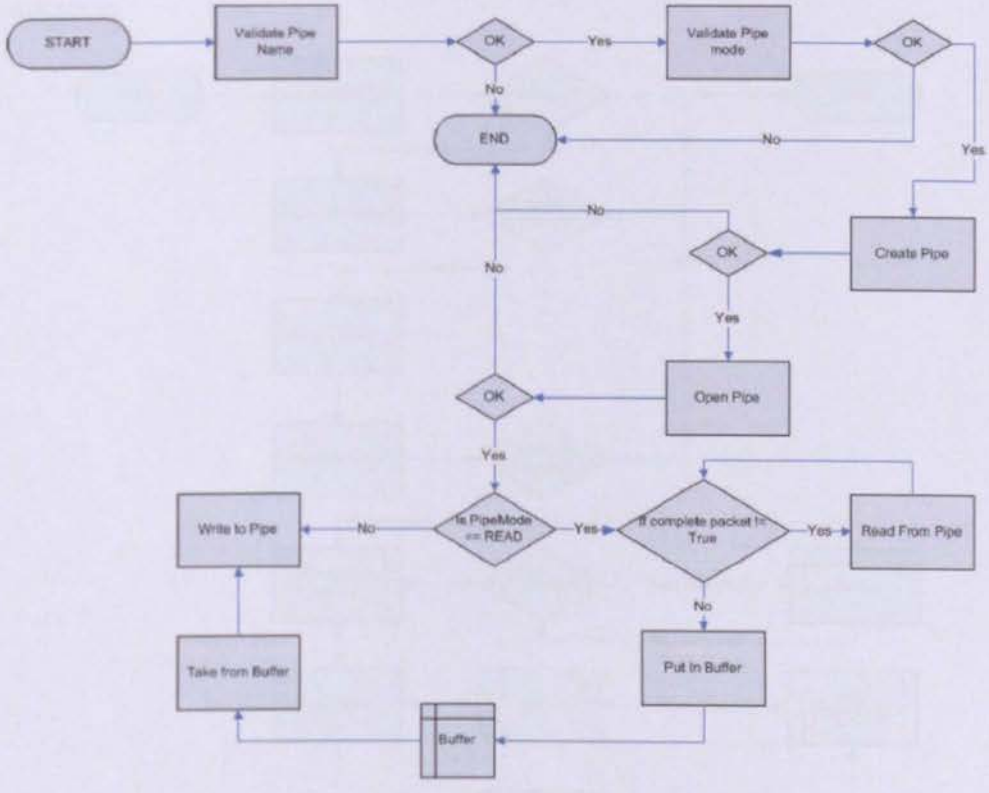

Figure 113 Flowchart of the Inter Process Communication (IPC)

6.3.3 MPI Interface

The MPI interface is the core aspect of the IB Resource Adaptor. It is the process that places the data into a virtual shared memory and also manages the memory window. Using a zero copy design, data can be moved from nodes to nodes without the expensive copies that exist in the traditional Message Passing form of communication (Ethernet). The MPI Interface starts with an initialising call, MPI_INIT(), the first MPI routine called in each process in order to establish an environment necessary for MPI to run. This environment may be customised for any MPI runtime flags provided by the MPI implementation. MPI_Win_Create() is a function in the MPI2 library that specifies a window accessible by remote processes which can subscribe to the window accordingly to access data using RDMA, (*see Figure 114*). The functions MPI_Put() and MPI_Get() of the MPI2 library triggers the RDMA read and write which are both based on the zero copy model of communication.

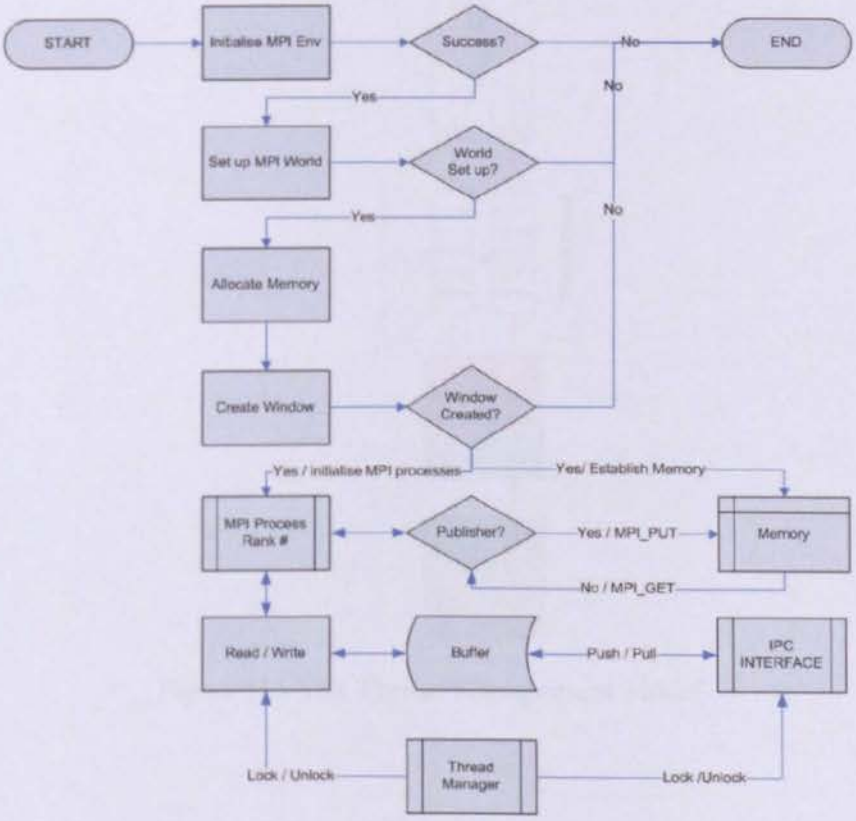

Figure 114 Flowchart of the MPI Interface

6.3.4 Thread Management

The Thread Manager handles the concurrent activities in between the MPI interface and the UNIX IPC Named Pipes. This is required since both the IPC Pipe Interface and the MPI Interface access data from a common buffer. In the SLA enforcement case study, we implemented a simple implementation of the pthread library using two mutexes to lock and unlock the buffer space when one the client accesses data. Note that the buffer that exists between the MPI Interface and the IPC Interface is handled by one thread wherein each MPI process has its own buffer and thread manager. The diagram in Figure 115 shows the thread handling of the buffer for one process mapped to one memory partition, which is identical to other processes mapped to their respective memory partition.

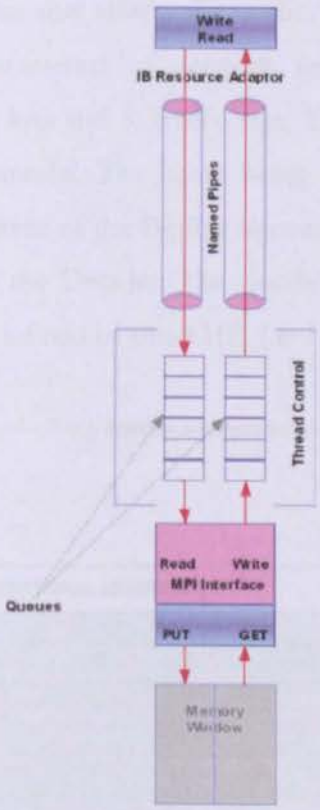

Figure 115 The Thread Management Model

6.3.5 InfiniBand's Virtual Lanes Management

The application of InfiniBand was not only advantageous for providing quality features such as performance and manageability of distributed nodes, but also on the management of traffic flow. We were able to exploit this feature and integrate the BipRyt algorithm for the primary purpose of managing these virtual lanes as per message requests (traffic flow), message types (premium messages) and consumers Service Level Agreement. InfiniBand provides a unique model of virtual lanes to manage traffic control of a system. The concept is based on adjusting the size of the virtual lanes, increasing and decreasing the bandwidth depending on the quality attributes that affects the traffic. The quality attributes are the five common Quality of Service parameters of network systems namely, 1) congestion, 2) throughput, 3) latency, 4) packet loss and 5) traffic rate. The quality attributes are evaluated by the designer using the AHP model. The latter, being a quality priority evaluator list, is provision to the Decider component of the BipRyt algorithm. The Perceiver is instructed to measure the QoS and feed it to the Decider. The decider then adapts the virtual lanes to preserve the quality attributes as defined by the AHP, (*see Figure 116*).

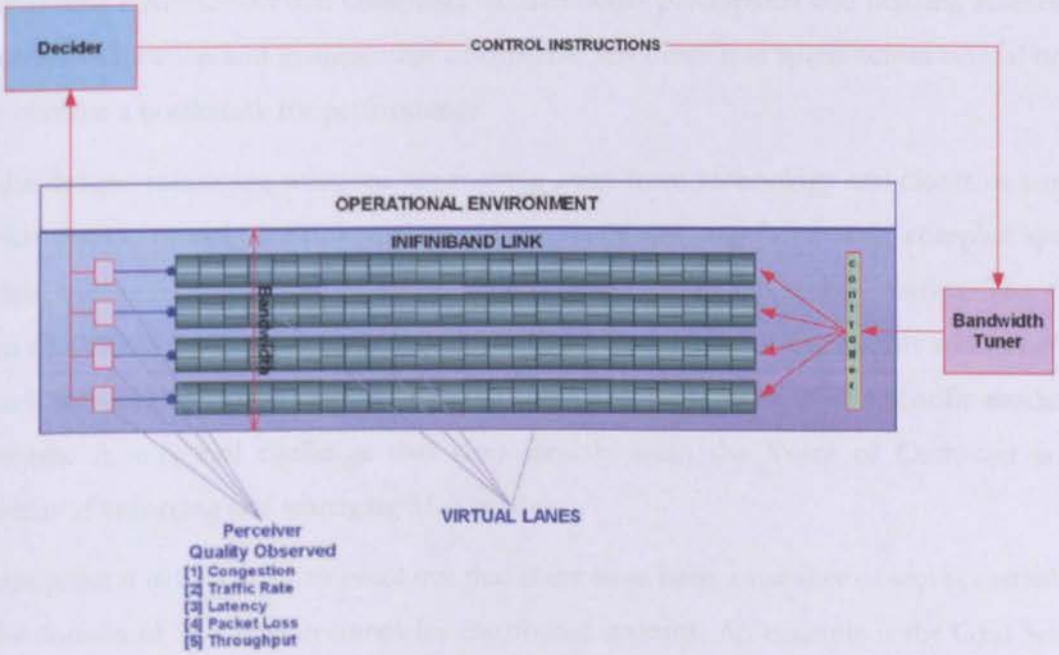

Figure 116 Integration of BipRyt within IB's Virtual Lanes

6.4 Service Level Agreement (SLA) in Telecoms

This section considers the application of distributed computing techniques to mobile telecommunications and wireless messaging infrastructures particularly to Service Level Agreement (SLA) enforcement (Mak07). It presents novel distributed techniques based on the application of InfiniBand technology over the RDMA channel Interface to develop a distributed Service Level Agreement and enforcement solution for the generalised mobile messaging infrastructure.

The motivating factor behind selecting the SLA enforcement problem as a case study is due to the fact that SLA are becoming extremely important in the Telecoms as the industry shifts from technology to services (*see chapter 2*) since QoS is a determining factor to ensure the viability of the business.

Fundamentally, SLA is a contract between suppliers and clients. In the domain of information system, the operation and management of SLA requires data integrity (Debu03). It is a complex and challenging problem to manage Service Level Agreement (SLA) within an operating environment that comprises of distributed participants and utilizing distributed resources. Allocation and management of dynamic resources that spans across several nodes may become a bottleneck for performance.

As the designs messaging solutions are moving away from technology and closer to a more service centric model, decision makers of the Telecoms are faced with complex quality models, quality of service and SLAs between stakeholders of the mobile market. The value chain of Telecoms operators now lies on the distributed nature of the market and to survive in such terrain, the classical models need to adapt to a newer and more scientific modelling approach. A very real challenge that rises directly from the Voice of Customer is the problem of enforcing and managing SLA models.

At this point it is important to point out that there have been a number of works carried out in the domain of SLA Enforcement for distributed systems. An example is the Grid Service Broker which is a project developed as part of the GridBus Programme (Buyy04). It

arbitrates collaboration between distributed participants as the broker identifies data sources for a given scenarios, e.g. an analysis or report. Suitable computational resources are discovered for specific job requirements. The broker provides a dynamic parametric programming model for writing grid applications (Buyy04). GridBus targets a higher degree of details about available resources (machine level), jobs and files which inherently makes it less scalable in large environments.

Another example is the Maui scheduler for network clusters which enforces SLA driven scheduling schemas (Clur05). It functions as a SLA engine for managing and allocations of resources to processes, whilst optimizing the resources consumption. The Maui achieves optimization in influencing processes, nodes, composite objects, QoS (CTQs), and policies.

Policy based scheduling of grid-enabled resource allocation was also presented by In et al (In04). The main difference with this approach is that the authors do not consider a centralised point of reference to SLA Enforcement. Policy based scheduling controls the request assignment to resources by adjusting resource priorities. Secondly, it efficiently manages resources in allocating usage quotas to target users. Thirdly it provisions for reservation based grid resource allocation.

In our attempt to resolve the problems inherent in enforcing the conflicting quality models of robust SLA enforcement, i.e. data integrity against performance, we understand that there are two major drivers of this study and they can be defined in two dimensions, which maps directly to the two types of pressures influencing the Telecoms business, covered in chapter 2.

Service Centric

As explained in chapter 2, the IP convergence with Telecommunication has changed the landscape and the value chain of the mobile market. As the latter grows to become more service centric, Quality of Service and enforcing SLA models are essential for service longevity.

Distributed Systems

As the mobile market gains momentum, the technology involved in accommodating the mobile services has to be designed for capacity. This means that resources can be easily and dynamically added to match demand. In this situation monolithic architectures fail, thus

designers opt for the deployment of distributed models. Distributed systems in the messaging infrastructure are becoming a natural evolution. The emerging complexity exists when attempts are made to apply robust SLA models over a distributed system, as this study will illustrate.

6.4.1 Quality of Service (QoS) and Service Level Agreement (SLA)

SLA and QoS are two words among the other quality models that are required and becoming more stringent in the Telecoms environment. In the past, SLAs were between operators and suppliers but now they are amongst multiple stakeholders. This implies that the process of enforcing SLA models becomes more complex yet more critical to manage.

As explained earlier, Service Level Agreement is a formal agreement made between two or more parties: the service provider and the service recipient. The SLA defines the basis of understanding between the two parties for delivery a service to an agreed level of quality. Services use different collaborations between system resources and when analysing the collaborations or resource interactions, we observe that SLA issues can arise at multiple levels: 1) SLA enforcement where operators may agree to enforce the SLA under which resources are made available to consumers and 2) SLA management, where consumers may want to access and interpret SLA statements published by providers, in order to monitor their agreements and guide their activities. Both providers and consumers want to verify that SLA protocols are applied correctly. In the context of the study, we focus on the element of SLA enforcement.

In classical message gateways, under a centralised server solution it is possible to throttle message rate for individual clients to prevent system flooding or to control the level of service offered to individual clients. This is considerably harder to develop on a distributed server solution since the client's messages may be spread over several dispersed nodes. An example of an SLA is the quota of messages, a given customer, is allowed to submit to a system. The quota can be a fixed number (e.g. 10000 messages limit) or a throughput limit (e.g. 500 messages per second).

Typically the distributed system consists of a number of heterogeneous nodes where individual clients may randomly send messages to any one of them. In our case study, there is an obligation to manage their quota across all the servers. Since the system is distributed, the issue of quota control enforcement is more complex; due to the fact that there is no single point of reference. A client may have several accounts and when a client submits a message, the account ID uniquely identifies the quota limit associated to that client. As message is submitted, the number of messages (balance) is incremented. The counter is updated on all the nodes of the distributed systems which results to all them having same counter for each accounts.

6.4.2 SLA Design & Architecture

In Figure 117, we illustrate a high level architecture of the SLA problem to enforce a quota management policy for the prevention of system flooding within a distributed messaging system. It provides a general view of how the SLA Enforcement application inter-communicates across distributed nodes.

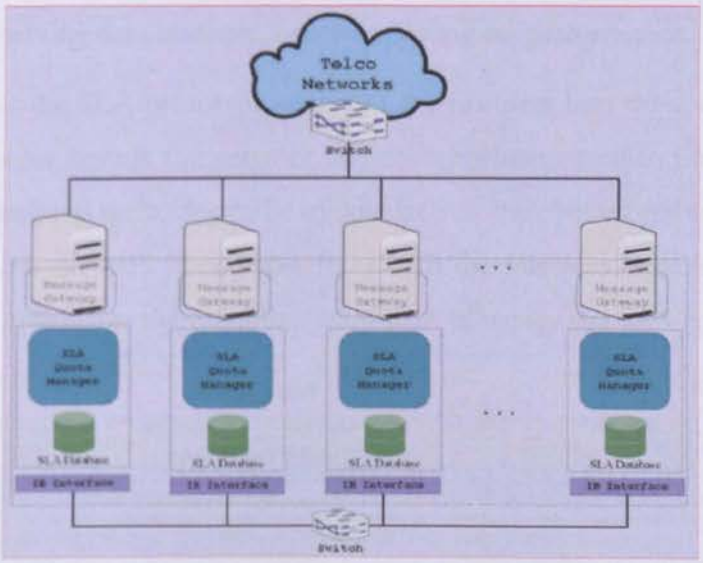

Figure 117 The SLA High Level Model

A client subscribes to the service provider (operator), opens an account and sends messages to the message gateway. The message gateway will query the SLA quota manager to confirm whether the message can be accepted for further processing. The SLA enforcement application consults a cache to validate the message and sends back the appropriate

acknowledgement to the message gateway. It then increments the balance for that particular account (client) and distribute the update to its neighbours. Each Message gateway on the network receives the update resulting to global SLA database synchronization. The complexity of this problem can be addressed if we look at two quality attributes, namely performance and data integrity.

- Performance

Through empirical observation we know that a typical message gateway can receive up to 3000 SMS per second in the UK. This implies that the number of quota update is $\frac{l}{t} \times (n-1)$, where l *is* the maximum number of messages, t is time (second) and n is the number of nodes which forces considerable loads on the interconnectivity.

- Data Integrity

To ensure that the data integrity amongst the distributed SLA database is high i.e. the variation of distributed values is low, designers implements poka yoke (Pres05) at run time. These are small modules for mistake proof and inevitably consume CPU resources during the process of preserving data integrity, hence impacting on performance.

The architecture of the SLA prototype separates the problem into three level of design (*see Figure 118*). The lower layer is the network technology which based to our evaluation study relating to the InfiniBand technology; the middle layer is the operational environment of the network upon which the MPI library interfaces with the business logic. The upper layer is the business logic, relating to the functionality of SLA counters and SLA conditionals.

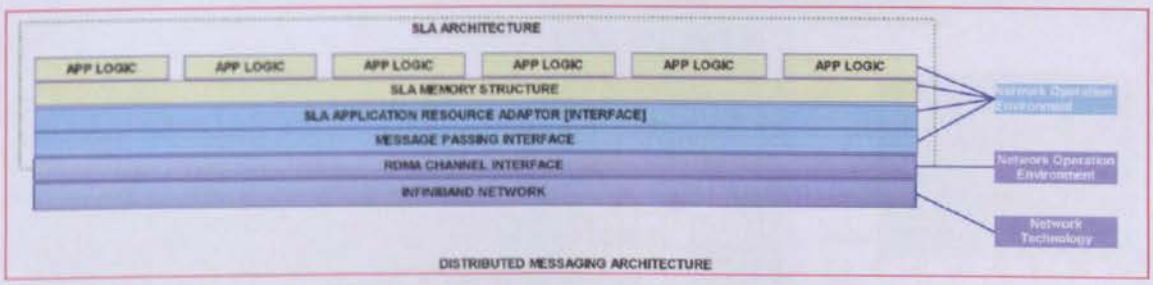

Figure 118 SLA Architecture

Since the problem involves conflicting quality attributes, we are required to use a different type of modelling techniques rather than just classical software modelling tools. Hence, we

use the concept of the **Blended Modelling approach** as presented in chapter 3, to explicitly capture and/or invent the functional and non-functional requirements (CTQs) of the SLA enforcement process in terms of requirement styles. As part of this approach, classification of requirements into specific styles constitutes an important phase, since the process enabled us to separate the requirements with respect to their nature and characteristics, hence providing us guidelines to understand which modelling tools and techniques to apply to which style of requirements. As such, we were able to use the classical requirement modelling tools, but we complemented it with value analysis tools such as HoQ, CTQ flow down, AHP and dynamic modelling tools such as prototyping.

The next sections illustrate the Blended Modelling Approach and development process of the SLA enforcement problem by showing the work done for each distinct requirement style i.e. 1) the quality attribute styles; 2) the functional and behavioural styles; 3) the data & structural styles and 4) the communication styles (dynamic).

6.4.2.1 Quality Attribute Styles

A set of requirements were drawn to drive the implementation of the SLA Enforcement solution. We adopted the QFD methodology and used HoQ matrix to plan and track the functional requirements and map each of them to the CTQs, (*see Figure 119*). Such exercise provides an insight to how important or critical a distinct CTQ is for distinct functions. As a result a bar chart is generated that prioritises the quality attributes in an order of importance.

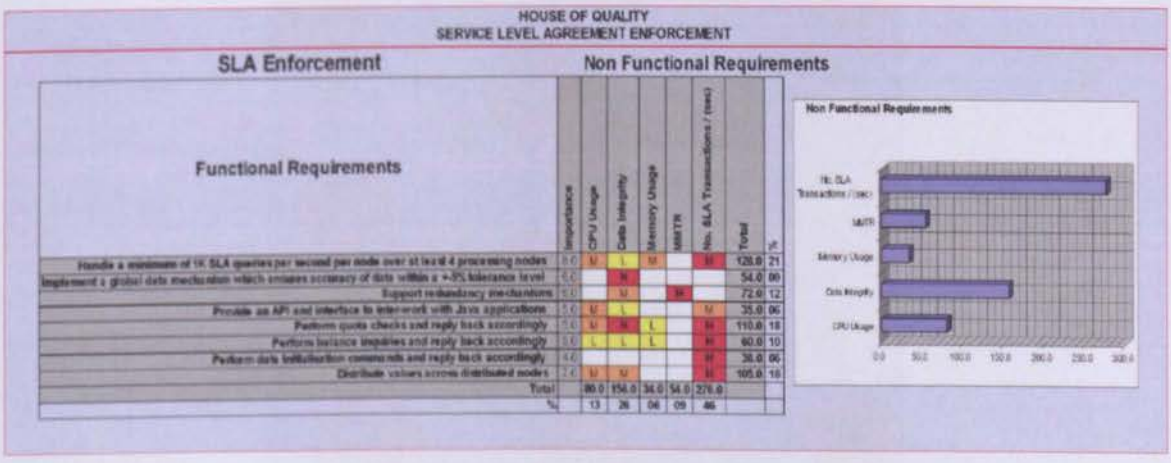

Figure 119 HoQ of SLA Requirements

The value of each CTQs presented on the House of Quality matrix are generally based on business knowledge and are assigned subjectively. This exercise often involves brain storming session amongst the business stakeholders, subject matter experts, marketing and product management. For the prioritisation and the evaluation process we complemented the HoQ with an AHP, which enables more formality and rigour in the evaluation process and provides an indication of the reliability of the measurements system (called consistency index). The bar chart (*see Figure 119*) shows the emergence of three most important CTQs namely, performance, data integrity and efficiency. The case study is driven to run simulation and prototyping models, in order to confirm facts about the functions of the SLA enforcement against the three CTQs. In obtaining a quantitative evaluation of the requirements, we move to model the functional design that defines the logic of the SLA solution.

6.4.2.2 Functional and Behavioural Styles

For the functional style, we used the requirement map model and flowchart diagram which is built on the knowledge gained from the HoQ. We designed a requirement map model to explicitly show the interactions between requirements and the type of relationships between each requirement (*see Figure 120*). It shows inter dependencies of requirements, hence highlighting the complex ones.

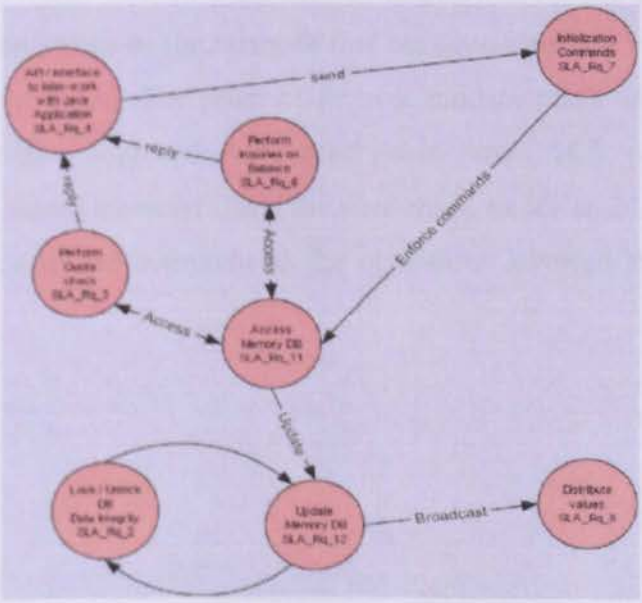

Figure 120 The SLA Requirement Map

To illustrate the functional description of the SLA enforcement problem, we designed a flow chart diagram (*see Figure 121*). The latter specifies the flow of process (action definitions) of the SLA enforcement function or mechanism.

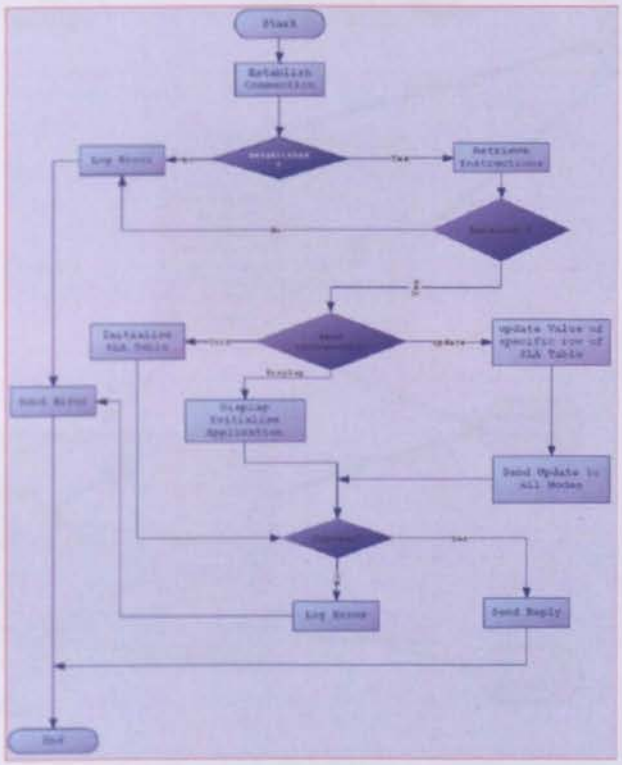

Figure 121 Flowchart of the SLA System

For the behavioural style, we used a state chart diagram as shown in Figure 122, which illustrates the state transitions of the methods that has been classified into entities, that later will become the distinct class. The point of focus is fundamentally on the change of state triggered by the locking and unlocking mechanism when SLA records are accessed vigorously from the shared memory. Using the state chart, we are enabled to statically model the dead lock situations and comprehend the operations involved that could resolve to deadlock situations.

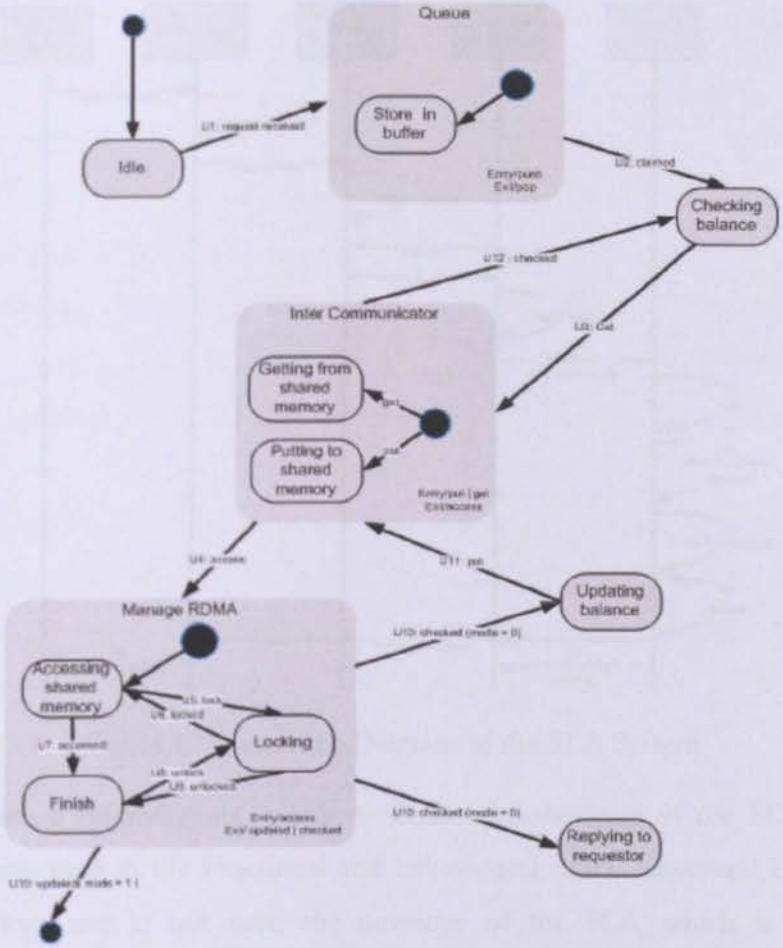

Figure 122 State Chart of the SLA System

6.4.2.3 Data and Structural Styles

Prior to defining the structure of the SLA model, we propose a sequence diagram that links the behavioural model of the state chart to a structural definition. In other words, a sequence diagram acts as a bridge between the functional and behavioural styles and the structural styles. The diagram in Figure 123 shows a sequence diagram that has been derived from the state chart and expresses the events, event handlers and procedures (methods) that interconnects the entities or classes stating the order of sequence.

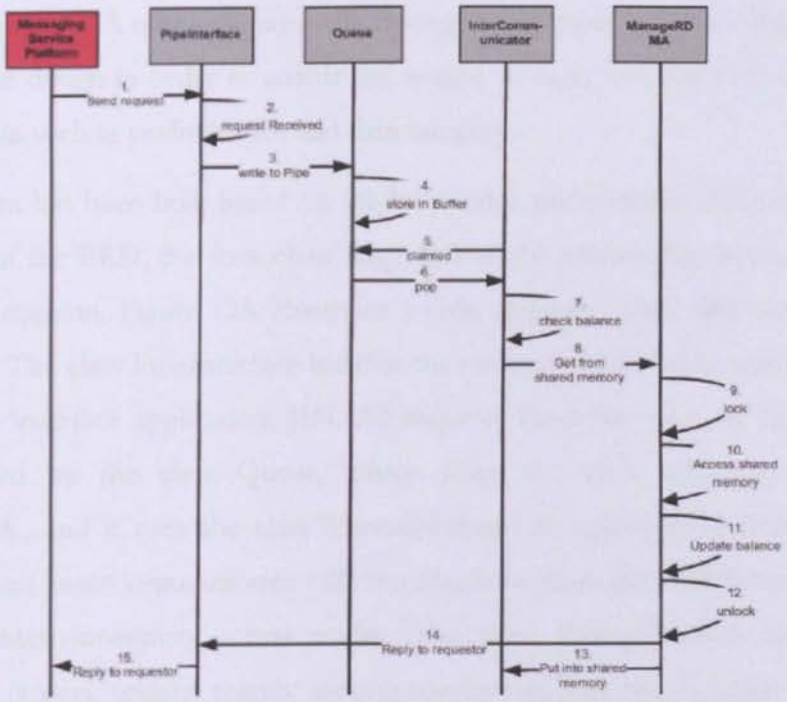

Figure 123 Sequence Diagram of the SLA System

From the sequence diagram, one is able to define the structure of the SLA model from knowledge represented in the functional and behavioural styles. Structural Design helps to build an ontology and in our case, the ontology of the SLA, which is dynamic, thus providing a Meta description of the data.

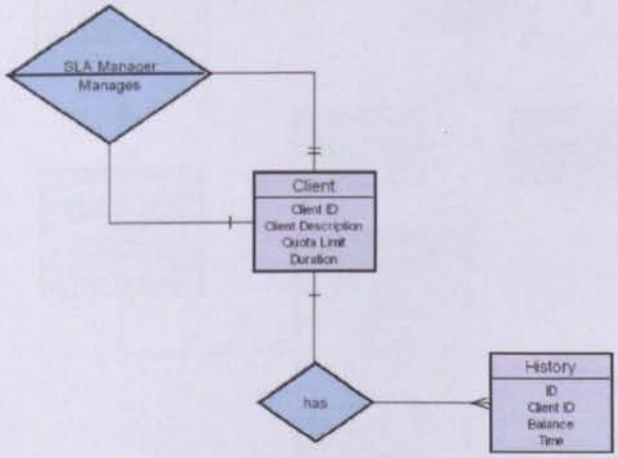

Figure 124 Structural Model of the SLA System

The diagram in Figure 124 shows an entity relationship diagram of the SLA enforcement problem which distinguish between the actual data and the process of treating the data. The entity Client has an attribute "duration" which defines the throttling pace, e.g. if the duration

is set to 0 then the SLA quota manages the throughput in a second. An entity History is also included in the design in order to enable the system to carry out statistical analysis on SLA application data such as performance and data integrity.

A class diagram has been built based on the knowledge gained from analysing the structural organization of the ERD, the state chart diagram and the relationship between classes from the sequence diagram. Figure 125, illustrates a class diagram of the prototype for the SLA Enforcement. The class PipeInterface handles the methods which send and receive data to/ from the java interface application, JIN. All requests from the pipe are buffered into the queue depicted by the class Queue, which links the class interCommunicator with ManageRDMA., and it uses the class ThreadManager to concurrently manage these two classes. The class InterCommunicator calls the functions from the MPI library to initiate and operate the interconnectivity across nodes. The class ManageRDMA handles the data manipulation, (access, update, search, locking mechanism and remove) that are required to manage the SLA records in the distributed shared memory.

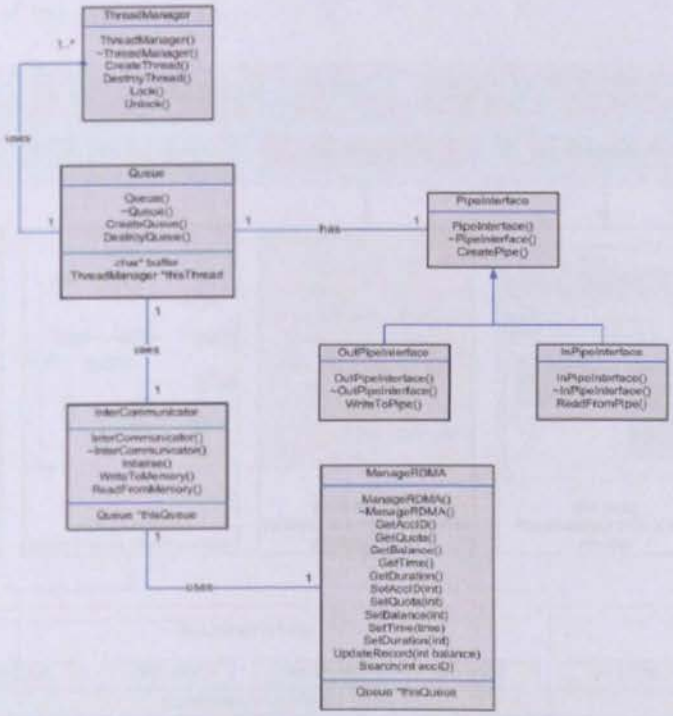

Figure 125 Class Diagram of the SLA System

6.4.2.4 Communication Styles

For the communication styles, we started by producing a number of Petri Nets simulation to validate any potential dead lock situation during SLA updates (Mak08). We designed a c++ prototype, (implementing the IBA resource adaptor within the grid architecture) using the structural and behavioural models of the requirements and the observations gathered from the simulation exercises. The prototype was exercised to validate its operations against the CTQs devised in the quality attribute style. This is further elaborated in section 6.4.3.

The modelling phase of the SLA problem consisted of several types of modelling disciplines summarised in Figure 126, which represents an application of the Blended modelling approach, covered in chapter 2. In Figure 126 we classified each requirement of the SLA into distinct styles. We analysed each styles to decide which type of modelling technique will best describe the requirement. Having devised several models, we then use the SLA architecture as a blueprint to guide us so that we are able to place and translate each model to the correct part of the architecture.

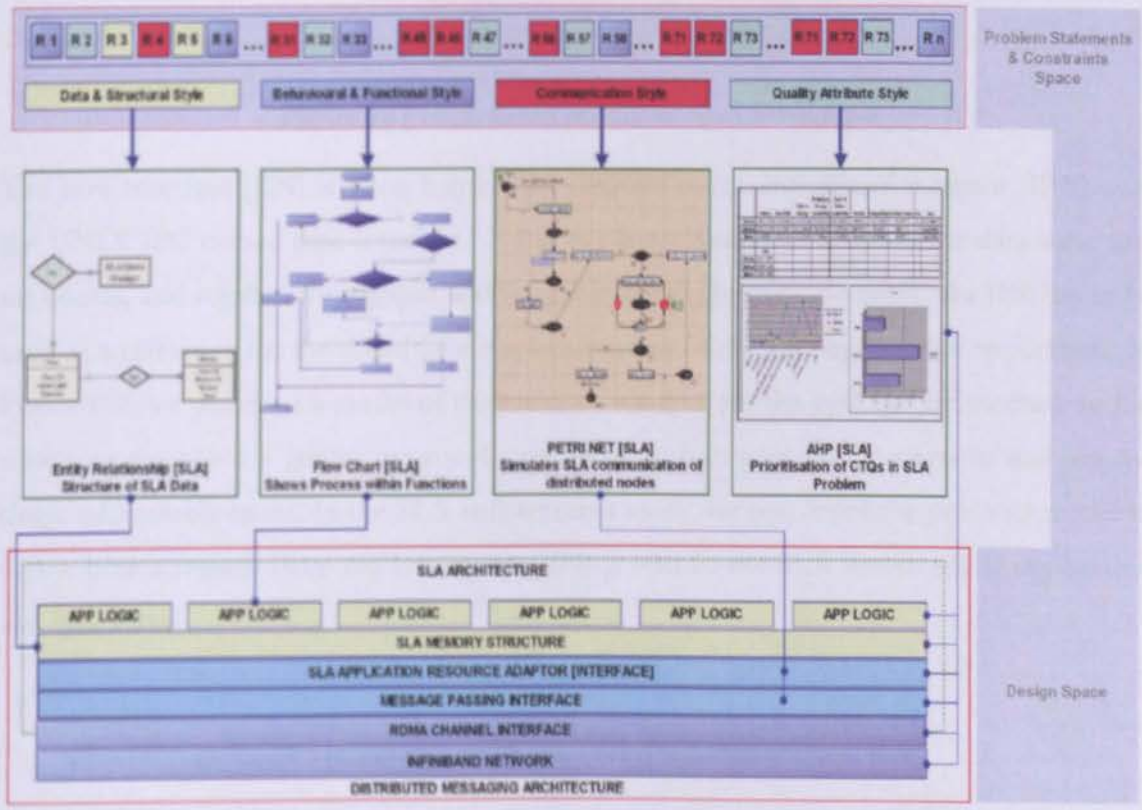

Figure 126 Blended Modelling Approach Applied to the SLA System

6.4.3 SLA Implementation

Based on the knowledge conceived during the design phase, a prototype was developed in C++. The MPI library was implemented to achieve platform independency since the code base is required to run on both InfiniBand and Ethernet network technologies for the purpose of comparative analysis. Earlier, we introduced the concept of InfiniBand Resource Adapter, (IBA) and in this section we situate the problem of SLA enforcement within the IBA environment. We built a Java Interface (JIN) to emulate the functionality of external messaging service applications for experimental purposes (*see Figure 127*).

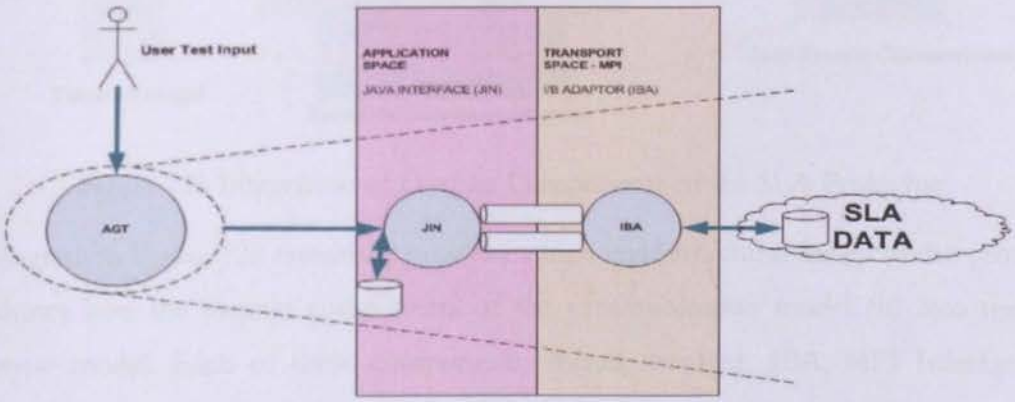

Figure 127 High Level Model of SLA Prototype

The Java Interface (JIN) is a test harness that reports to the InfiniBand Adaptor (IBA) over the UNIX IPC named pipe interface. It has the functionality to initialise the data structure, set quotas, and regulate throughput and output statistics. In our case study, the JIN has to be used as a reference for the actual Java implementation of the message service application. In Figure 108, we provided a model of the memory window for the grid like architecture within which we proposed a generic protocol structure that transport IBA packets in and out the single addressable space. In the SLA enforcement study, we employed the protocol model to define how a request from the Java space (JIN) is sent to the SLA Enforcement application and vice versa.

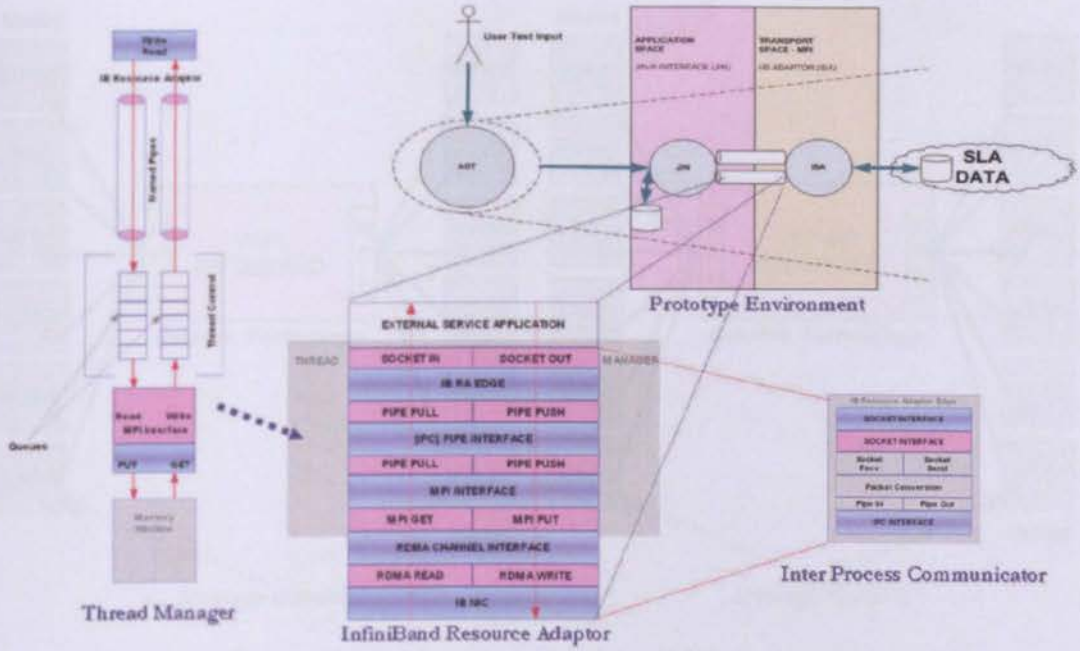

Figure 128 Integration of Distinct Components of the SLA Prototype

The diagram in Figure 128 essentially provides a high level structural design of the prototype and shows how the distinct components of the communication model fits into the SLA prototype model. Each of these components: thread manager, IBA, MPI Interface and UNIX IPC named Pipes have been explained in earlier sections.

6.4.4 Test Environment & Observations

The primary objectives of the prototyping tests are as follows:

- To compare the throughput performance of the SLA quota manager when deployed on InfiniBand against Ethernet.
- To compare CPU efficiency of the InfiniBand against Ethernet deployment.
- To check the data integrity of the quota and balance of the SLA Table on all the nodes and compare the results of InfiniBand and Ethernet.
- To check the speed ratio of the distributed messaging system using InfiniBand.

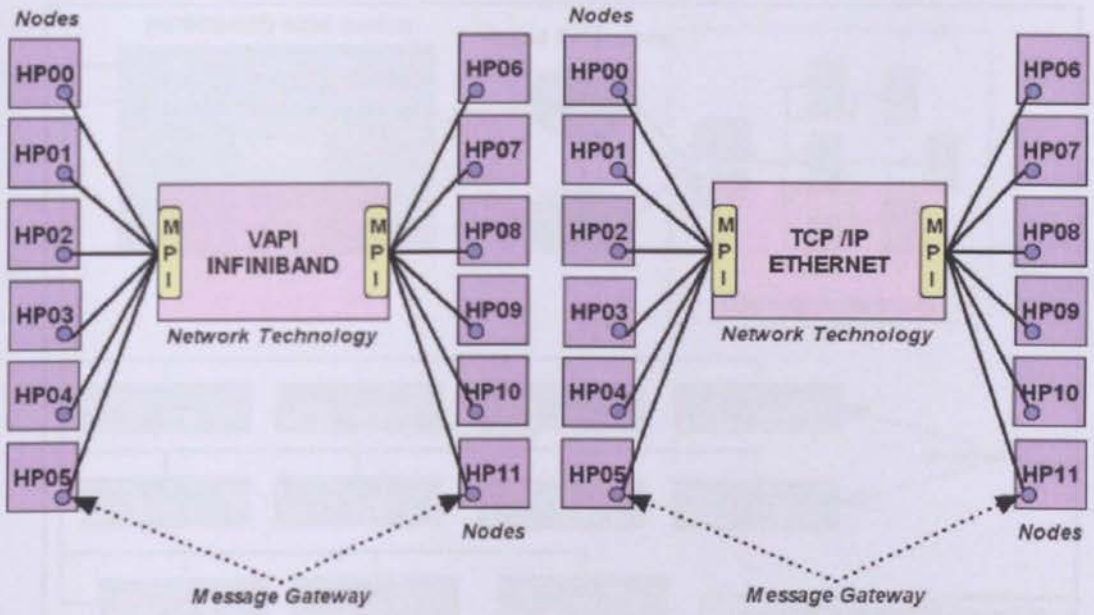

Figure 129 Environment Model of SLA Prototype

In essence, the aim of the experiments is to compare the difference in performance, efficiency in terms of CPU usage and data integrity of the SLA applications over two distinct network technologies, namely InfiniBand and Ethernet. The benchmark against each of these CTQs will be running the SLA Enforcement prototype over IB (VAPI) and Ethernet (TCP/IP). For each message service application or message gateway as depicted in Figure 129, the user input can be injected for various testing scenarios. The Java Interface (JIN) provides the interface and input regulator, and an InfiniBand Resource Adaptor (IBA), providing access to the shared memory based on the grid architecture. The networking environments of the prototype test bed are illustrated in Figure 129 where HP00 ... HP11 represents the host names of the test servers.

6.4.4.1 Prototype Test bed

In Figure 130, we illustrate the platform architecture for the tests carried out in the SLA experiment. 11 *HP Proliant DL 385* servers are connected with InfiniBand cables via an *InfiniBand 9096 switch* (Volt05). The switch is connected to two redundant load balancers that distribute the message load from external client networks to the internal system. This particular hardware solution was design to install, configure and deploy the SLA Enforcement prototype.

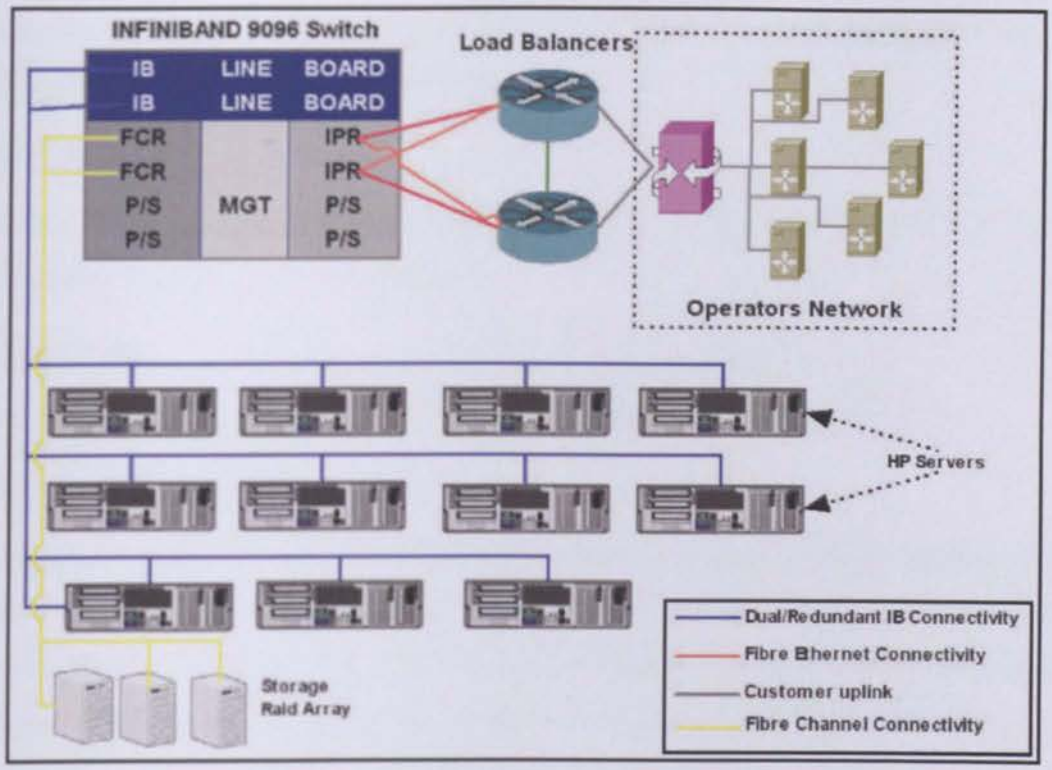

Figure 130 Platform Architecture of SLA Model

The test bed consists of a clustered system with *HP Proliant DL 385* servers. Each server has dual *AMD Opteron 2.60 GHz* processors with *8 GB* memory. The machines are connected by *Mellanox InfiniHost Dual Port 4X HCA adapter* through an *IS R9096* InfiniBand Switch (Mell05), wherein the *HCA* adapters work under the *PCI-X 64-bit 133MHz* interfaces. We used the *RHEL 4 U1* operating system with *2.6.9-11 kernel* and finally the compilers used were *GNU g++* and *java 1.5*.

6.4.4.2 Observations

The objective of exercising the prototype is to validate the behaviour of the model against the predefined CTQs and the following explains the procedures of observation:

Performance Analysis

To compare the performance (no. of SLA updates over time) between InfiniBand VAPI protocol and Ethernet TCP IP

Efficiency of System

To compare the CPU usage when deploying the SLA Enforcement prototype over InfiniBand VAPI against Ethernet TCP/IP

Data Integrity

To check the data integrity of the SLA Enforcement prototype as the number of nodes (mpi ranks) increases in the distributed system.

Speed Up Ratio

To check the speed up ratio of the SLA update throughput over InfiniBand.

Performance Analysis

The graph in Figure 131 compares the performance of the SLA Enforcement application when deployed on InfiniBand VAPI and Ethernet TCP/IP. The graph addresses the performance as number of SLA transactions per second, i.e. the number of request being replied and the number of updates being published over the shared memory space.

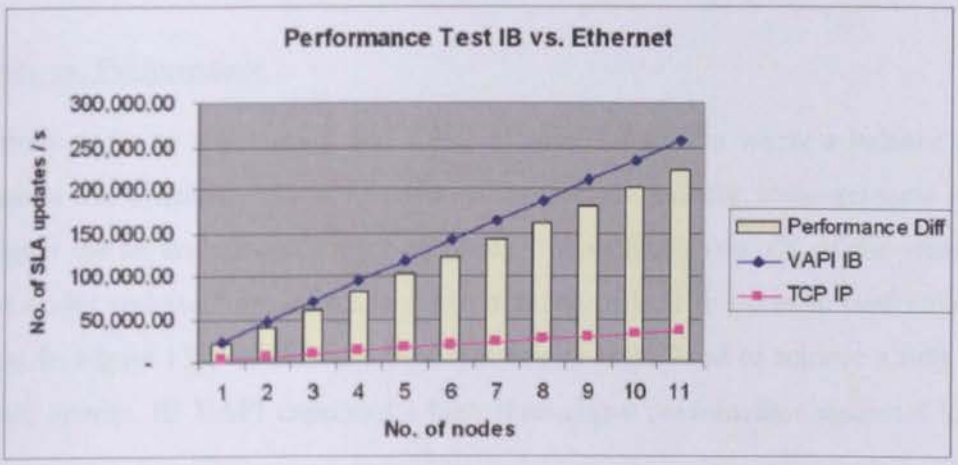

Figure 131 Performance Analysis of IB against Ethernet

The graph shows a clear gap between TCP/IP over Ethernet and VAPI (Verbs API) over InfiniBand in terms of throughput performance. We demonstrated that InfiniBand out performs TCP/IP by a factor of 7.23.

Efficiency of System

The diagram in Figure 132, shows a graph of the SLA Enforcement application running on *11 nodes* deployed on InfiniBand VAPI and Ethernet TCP/IP. The graph clearly illustrates that when using InfiniBand VAPI, the CPU usage and consumption is 3 to 4 times less than

Ethernet TCP/IP. This is because when TCP/IP packets are transported, they are required to go through the TCP/IP stack with Kernel calls which consumes CPU time. On the other hand IB VAPI uses RDMA concepts that bypass the expensive Kernel calls (chapter 3).

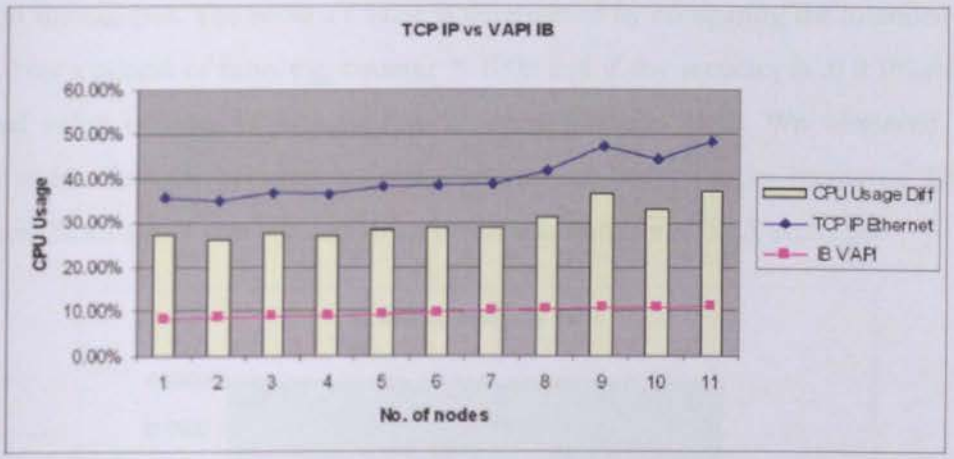

Figure 132 Efficiency Comparison between IB and Ethernet

Efficiency vs. Performance

The primary order to scalability is the ability to model a system where a balance between performance and efficiency exists. In order to achieve the balance, measurements are done and designer use an analytical approach to change the variables (the x's) of the system, such as no. of nodes and attempts are made devise the line of best fit between performance and efficiency. In Figure 133, we illustrate the capability of InfiniBand to achieve a fully scalable distributed system. IB VAPI expresses a high throughput performance against a low CPU usage.

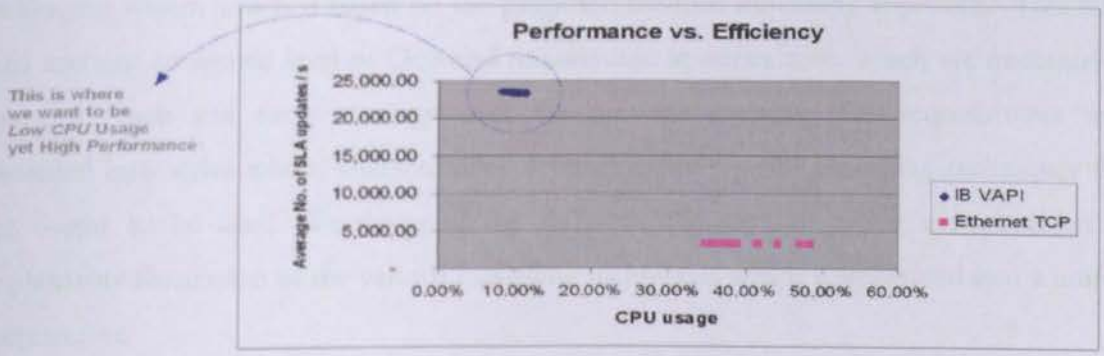

Figure 133 Performance vs. Efficiency between IB and Ethernet

Data Integrity

The next graph explains the data integrity test carried out to find out level of accuracy in updating the SLA counter distributed across multiple nodes over a period of time with maximum throughput. The accuracy value is determined by comparing the intended value of counter over a period of time. e.g. counter = 1000 and if the accuracy is ± 0.1%.this means the actual value of counter observed is between 990 and 1010. We observed that the accuracy index deviates between ± 0.16% as the number of nodes increases. Each tests progressing from 1 to 4 corresponds to an increasing number of SLA accounts.

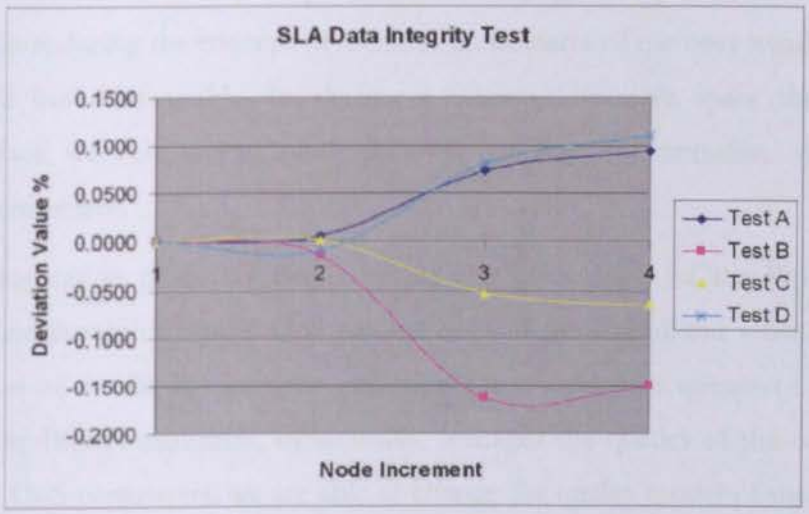

Figure 134 SLA Data Integrity Analysis

6.5 Conclusion

In this chapter we presented a framework for the development of a novel distributed messaging system which is based on the proposed blended modelling approach. This takes into account an agreed level of QoS and requirement specifications, which are necessary to transport each and every message over the message gateway. The requirements were classified into styles whose characteristics determine the type of modelling techniques that are ought to be used. We designed the different requirement styles and provided an explanatory illustration of the various modelling techniques which is integrated into a unified architecture.

The proposed communication model was inspired from the model of the human conversation, and based on its structural design (ERD), we constructed the communication as a grid architecture for connecting participants and messaging services of the distributed systems. The grid architecture is based on an advanced network technology called InfiniBand, exploiting the concept of Remote Direct Memory Access (RDMA) and distributed shared memory. It lessens the complexity involved in managing dynamic distributed systems, i.e. nodes joins and exiting the system, by managing the nodes and services, as an information system, which we represented as a database structure. We showed how the grid architecture address the problem of managing distributed nodes and horizontal scalability by introducing the concept of memory coordinates of memory windows which are not hardwired but configurable. In sharing a single addressable space through RDMA channel interface, we were able to mimic the manageability of a centralised system within a distributed architecture.

At the implementation phase we demonstrated the applicability of the BipRyt algorithm within the management of typical QoS parameters and the InfiniBand virtual lanes model. The integration of the BipRyt into the grid model lies within the queues of each memory window. As the BipRyt algorithm, dynamically, manages the queues of the communication model against QoS parameters, we are able to change the quality model of the system at run time, e.g. choose between throughput performance or robustness.

In order to test and validate the grid architecture and the operability of these components, we formulated experiments based on the problem of Service Level Agreement, in the Telecoms. Within this case study, we subjected the SLA enforcement experiments on both the InfiniBand and Gigabit Ethernet technologies, to compare CTQS such as performance, reliability, scalability and data integrity.

A prototype of the SLA enforcement application was implemented and exercised against the CTQs. Based on the observational results, we concluded that our framework operates very well in terms of performance, efficiency, data integrity and scalability. This implies that our developed system runs at very high performance, exploiting the quality of scalability from distributed systems yet, enabling a single point of management for configuration and provisioning, exploiting the benefit from centralised system.

7 CONCLUSION & FUTURE WORKS

7.1 Conclusion

The mobile market is expanding so rapidly that the value chain of the Telecoms operators has been broken down into a value web. What were once called gatekeepers, Telecoms operators, are now mere players amongst many stakeholders. Their business landscape has changed, since in this new value web, the returns are being distributed over a complex mesh of stakeholders. In order to keep the competitive edge, operators have to reassess their definition of mobile services and re-evaluate their technology.

The research work started by investigating the evolution of the mobile market and its impact on the operability of Telecoms carriers. We observed that operators are being pressured by two forces; a pressure from the outside, the growing demand from consumers for larger number of mobile services, leading to complex message transfers and a pressure from the inside, the strategic change of technology to enable these services. These two forces are interdependent and this can be explained through the current phenomenon of IP convergence. The latter is now impacting the technology used to enable these new breed of services, resulting in the problem of co-habitation (the old meets the new) to the domain of the Telecoms, since their native networking infrastructure has to blend with the IP networks. This is a fundamental aspect of evolution within the messaging system which brings us to the heart of this study.

For the Telecoms systems to evolve, they need to be operated on a distributed environment, which tends to naturally, conform to the requisites of building adaptive and evolutive systems. Distributed systems enable the logical abstractions of software components from their connectors, thus facilitating the structure to accommodate changes.

The aim of this thesis is to propose and validate the implementation of a new distributed messaging infrastructure that will sustain the dynamics of the mobile market by providing innovative technological resolutions to the common problem of quality modelling, communication, evolution and resource management, within the Telecoms. The shift from monolithic to distributed messaging systems, introduced new complexities to the problem domain of messaging, which were analysed and modelled.

To facilitate the analysis of the problems and their implications, we treated them as an information system, with the intention of exploiting the capabilities of structural organisation. We referred to the structure as a Class of problems, which is multidisciplinary and requires the use of multiple modelling approaches in order to solve the problem attributes. The modelling disciplines practiced were both, structural and dynamic as we learned that classical software engineering tools such as UML, ERD and SSADM, were too unvarying in nature, to model the dynamic character of the messaging problem. These tools are good candidates for modelling the deterministic part of a system, which conform to the character of the functional and structural requirement styles. But with emergent behaviour, some aspects of probabilistic and statistical modelling concepts are required. Probabilistic techniques lead to the discovery of predictive knowledge on the non-deterministic character of a system which is achieved through robust simulations and prototyping.

Therefore, we proposed a blended modelling approach which employs tools and methods aspired from both the structural and dynamic arenas. We devised a research process model to direct the development of the proposed distributed messaging system and this process is fundamentally based on the blended modelling approach. This enabled the translation of structural models to traceability matrices, such as HoQ and AHP, which in turn provides the interface to access statistical analysis of the models. Using a multitude of modelling techniques, we evaluated the different tools for design, simulation and implementation of distributed messaging systems. For example the evaluation of the simulation environment provided the evidence that Coloured Petri Nets measure up to the requirements of simulating the dynamic aspect of messaging systems.

In order to assess the level of QoS required in messaging and to investigate the potential use of distributed systems within a messaging environment, we considered a real life messaging systems (VMR), whose communication model was based on queues. Thus we simulated the

behaviour of several concurrent interconnected VMR queues using CPN by handling key parameters, such as queue's buffer population, performance throughput, packet loss and congestion. A clear definition of the quality attributes was achieved by examining the impact of the CTQs on the VMR's functions and how they conflicted amongst themselves. The functions were translated into Petri nets and we compared the arrival rate of messages into the queues against the departure rate, hence devising the transit time of a single message within the system. We also simulated the system to assess the buffer population of the queues given a fixed throughput of messages, which can provide designers with the ability to plan and regulate system capacity.

Moving from a monolithic design of the VMR, its agent model was migrated to a distributed system which required the simulation of various communication agreement models of the new design, ultimately leading to the concept of service discovery. We demonstrated how the various agreement models work across distributed nodes and how they influenced the choice of communication styles. The results of the experiments demonstrated that at various stages of system life cycle, different types of service discovery strategies have to be implemented in order to manage the nodes. To know which strategy to use, we had to look at the characteristics and CTQs which are essential for that particular communication agreement model. We have shown that this type of knowledge can only be obtained through dynamic modelling techniques and indeed such knowledge has been acquired from the simulation of service discovery. The analysis of the simulation results provided implementation directives and validation for the proposed distributed messaging systems.

The aspect of evolution, adaptability and resource management within the distributed messaging system were addressed by proposing a new multi-criteria decision making algorithm (BipRyt algorithm). The algorithm has been designed for the management, distribution, control and optimisation of systems resources (e.g. nodes, CPU, channel, memory etc). It is an empowerment strategy that provides autonomy to several parts of the systems and its novelty lies in the ability to use a multitude of quality attributes for decision making.

The algorithm is based on a statistical analysis of the quality model, and decisions are taken for the welfare or overall health of the system. During the implementation of the BipRyt algorithm, we observed that techniques such as the AHP and the HoQ provide instructions

that can be automated through a sequence of actions within the code base. Hence, the algorithm has been constructed to represent both Run Time quality assurance and dynamic quality enforcer. In addition to system related quality criteria, the BipRyt algorithm can take into account the user's perspective of the quality model, i.e. the user can configure the quality features that are expected from a system. It ensures the system follows the quality trend of the user, thus preserving the quality model against a potential decline in quality, especially within distributed systems. This is even more significant in the Telecoms, since the service node population and the value associated to a message, grows in accordance to market demand. We demonstrated that the BipRyt algorithm addresses the problem of Lehman's 7th law of evolution, Decline Quality, and the 8th law of evolution through the implementation of a continuous feedback system.

To validate the functionality of the BipRyt algorithm in enforcing a defined quality model, we presented a case study that implements the algorithm as a load balancing decision making mechanism within the communication model of distributed messaging systems. We tested the capabilities of the BipRyt algorithm against the Round Robin, Response Time and Least Connections strategies. The result showed that the BipRyt algorithm efficiently distributed the work load especially in heterogeneous network, with different capability servers, since it followed an economic model, balancing the demand for message processing with the capacity of servers (purchasing power).

Finally, we devised a framework for the development of a novel distributed messaging system which is based on the proposed blended modelling approach. The requirements were classified into styles whose characteristics determine the type of modelling techniques that are ought to be used. We designed the different requirement styles and provided an explanatory illustration of the various modelling techniques which are integrated into a unified framework. The communication model of the distributed messaging system was inspired from the model of the human conversation, and based on its structural design (ERD) we constructed the communication as a grid architecture. The grid architecture is built on an advanced network technology called InfiniBand, exploiting the concept of Remote Direct Memory Access (RDMA) and distributed shared memory. Such an architecture, lessens the complexity involved in managing dynamic distributed systems, i.e. nodes joining and exiting the system, by organising the nodes and services, as an information

system. We achieved effective management of distributed nodes and horizontal scalability by introducing the concept of memory coordinates of the memory windows which were not hardwired but configurable. In sharing a single addressable space through RDMA channel interface, we were able to "*mimic*" the manageability of a centralised system within a distributed architecture.

Moreover, the BipRyt algorithm was integrated within the communication model to intelligently manage the QoS parameters of interconnected queues of the grid architecture as well as to distribute the traffic flow of the InfiniBand virtual lanes; making the overall system adaptable to its environment. In order to test and validate the grid architecture and the operability of these components, we formulated experiments based on the problem of Service Level Agreement, in the Telecoms. Within this case study, we subjected the SLA enforcement experiments on both the InfiniBand and Gigabit Ethernet technologies, to compare CTQS such as performance, reliability, scalability and data integrity. A prototype of the SLA enforcement application was implemented and exercised against the defined CTQs. Based on the observational results, we concluded that our framework operates very well in terms of performance, efficiency, data integrity and scalability. This implies that our developed system runs at very high performance, exploiting the benefit of distributed systems (such as high availability and horizontal scalability) yet, enabling a single point of management which is established on a database management system, exploiting the benefit of centralised systems.

7.2 Future Works

There are three principle areas of research within this thesis that require extension for future works:

Firstly, we believe that the long term memory, (substratum), or storage is a very crucial and complex problem that is faced by the Telecoms. Although storage is outside the scope of this research, we believe that, since it is defined by the type of communication, storage is an issue that requires further research in the domain of distributed messaging system. A decision making algorithm based on the rules governing both the communication agreement model and the communication styles is an efficient way of attributing storage resources. As

we demonstrate through a simulation experiment in chapter 4, the communication medium is dependent on the communication styles and vice versa. Therefore there is a clear need for a decision making mechanism to select the type of storage for a particular message, based on the communication style. The BipRyt algorithm can be adapted to enforce such a decision process which can be founded on a knowledge based system (i.e. a history of previous storage events).

Secondly, the management of the communication model as a grid architecture leads us to the concept of Virtualization. Virtualization has the inherent feature of managing many as one, which relates to our concept of designing a distributed system which can be managed by a DBMS using RDMA. There are many examples of virtualization ranging from server consolidation, portable applications and workspaces to disaster recovery or hot standby. Virtualization provides benefits of offloading the application by moving the managerial tasks to the operational environment of the system. It has not been thoroughly assessed in this study but we believe that the natural progression of the grid architecture over RDMA is, its integration with the virtualisation model. We believe that the proposed grid architecture can be extended to interface with the operational capabilities of virtualization.

Thirdly, in our study, the communication model of the grid architecture was built using the MPI application protocol. However, we were faced with the inability for any MPI library to manage fault tolerance aspects and found out that many of the application protocol such as MPI, uDAPL, PVM, have limited fault tolerance mechanisms. These application protocols were essentially devised to run parallel jobs in bulk which fall in the category of intensive computing application. However in the Telecoms business, require a real time system wherein each message has a value attached to it, requires interpretation and robust fail over strategies. We believe that an extension work is needed on these application protocols to address the messaging requirements by focussing on the feature of exception handling for fault tolerance and high availability. Moreover an abstraction is required to build the "*catch up*" process, which essentially is a distributed storage mechanism of the communication history for failed nodes. The history can be retrieved accordingly, when each node comes alive again. We have presented such abstraction models in Petri Nets and the "*catch up*" process of the human conversation and these models can be implemented in the standard of these application protocols.

7.3 Summary

During his work on communication and language, G. Bateson claims that the word "communicate" is one of those rare words that do not have an antonym. Our endeavour to build software systems that handles the logistic of communication in the form of short messages, forced us to look outside the realm of technology and embrace concepts such as quality, voice of customers, voice of the market, service longevity and development process; as we realized, these are attributes that belong to humans rather than to machines. Technology alone did not solve the problem, as the class of problem we put forward, proved to be multi-disciplinary at technological, cultural and social level. Being social, it grows, evolves and soon the complexity of the system is unmanageable leading to a decline in quality and the communication breaks down. Communication is not about software or machines but about us, humans. Since the dawn of mankind, we are driven to search for ways to improve upon shouting; a curse or a blessing, tomorrow will decide.

"beep...beep...beep..."

Sputnik

REFERENCES

(Jack57) Jackson, J. R. "*Networks of waiting lines*", Operational Research 5, pp. 518-521, 1957

(Pet62) Petri C A, "*Kommunikation mit Automaten*", PhD thesis, Institut f¨ur instrumentelle Mathematik, Bonn, 1962

(Gor67) Gordon, W. J.; Newell, G. F, "*Closed queueing systems with exponential servers*", Operational Research 15, pp.254-265, 1967

(Sche67) Scherr, A L, "*An analysis of time shared computer systems*", MIT Press, Cambridge, Mass., 1967

(Sim69) Simon H A, "*The sciences of the artificial*", Cambridge, M.I.T. Press, 1969

(Bat71) Batteson G, Houseman M, Severi C, "*La cérémonie du Naven*", ed. Minuit 1971

(Buz71a) Buzen, J P, "*Analysts of system bottlenecks using a queueing network model*", in Proceedings of ACM SIGOPS Workshop System Performance Evaluation, ACM, New York, pp. 82-103, 1971

(Buz71b) Buzen, J. P. "*Queuemg network models of multiprogramming*", PhD Thesis, Div. Eng. and Applied Physics, Harvard University, Cambridge, Mass, 731 575, 1971

(Chai71) Chaitin G J, "*Computational Complexity and G¨odel's Incompleteness Theorem*", ACM SIGACT News, No. 9, IBM World Trade, Buenos Aires, pp. 11- 12, April 1971

(Jung71) Jung, C. G, "*The collected works (Bollingen Series XX)*", R. F. C. Hull, trans.; H. Read, M. Fordham, and G. Adler, eds. Princeton: Princeton University Press, 20 vols, 1921 - 1971

(Moor71) Moore, C G III, "*Network models forlarge-scale time sharing systems*", Tech.Rep. 71-1, Dept. Industrial Eng., Univ. Michigan, Ann Arbor, PhD Thesis April 1971

(Hugh73) Hughes P H; Moe G, "*A structural approach to computer performance analysis*", in Proceedings of AFIPS National Computer Conf, Vol. 42, AFIPS Press, 109-119, Montvale, N.J, 1973

(Buz73) Buzen J P, "*Computational algorithms for closed queueing networks with exponential servers*", Commun., ACM 16, 527-531, September 1973

(Bask75) Baskett, F.; Chandy, K. M.; Muntz, R. R.; and Palacios, J. "*Open, closed, and mixed networks with different classes of customers*", J. ACM 22, pp. 248-260, April 1975

(Buz75) Buzen J P, "*Cost effective analytic tools for computer performance evaluation*" in Proceedings of IEEE COMPCON„ pp. 293-296, IEEE, New York, 1975

(Klei75) Kleinrock L, "*Queuemg systems I*", John Wiley, New York, 1975

(Schu75) Schumpeter J A, "*Capitalism, Socialism and Democracy*", Harper, New York, 1975

(Boeh76) Boehm B W, "*Software Engineering*", IEEE Trans. Computers, pp. 1,226 - 1,241, December 1976

(Giam76) Giammo, T, "Validation *of a computer performance model of the exponential queueing network family*", Acta Inf. 7, 137-152, 1976

(Klei76) Kleinrock L, "*Queuemg systems II*", John Wiley, New York, 1976

(Rose78) Rose CA, "*Measurement procedure for queueing network models of computer systems*", Computer Survey 10, pp. 263-280, September 1978

(Leh80a) Lehman MM, "*On Understanding Laws, Evolution and Conservation in the Large Program Life Cycle*", Journal of System and Software, vol 1, no. 3, 1980

(Leh80b) Lehman MM, "*Programs, Life Cycles and Laws of Software Evolution*", in the Proceedings of IEEE Special Issue on Software Engineering, vol. 68, no. 9, Sept. 1980

(Saat80) Saaty T L, "*The Analytic Hierarchy Process*", New York, NY, Graw-Hill, 1980

(Boeh81) Boehm B W, "*Software Engineering Economics*", Prentice-Hall, Englewood Cliffs, N. J, 1981

(Holz81) Holzmann G J, "*PAN: A Protocol Specification Analyzer*", Technical Report TM81-11271-5, AT&T Bell Laboratories, March 1981

(Pete81) Peterson, J L, "*Petri net theory and the modeling of systems*", Prentice Hall, Englewood Cliffs, New Jersey, 1981

(Dole82) Dolev D, Strong R, "*Distributed Commit with Bounded Waiting*", in the Proceedings of the 2nd IEEE Symposium on Reliability in Distributed Software and Database Systems, July 1982

(Lamp83) Lamport L, "*The Weak Byzantine Generals Problem*", Journal of the ACM, vol 30, no 3, pp 669-676, July 1983

(Holz85) Holzmann G J "*Tracing Protocols*", AT&T Technical Journal, vol. 64, pp. 2, 413-2, 434, December 1985

(Leh85) Lehman M M and Belady L A, "*Program Evolution, - Processes of Software Change*", Academic Press, London, 1985

(Chan86) Chandy K M, Misra J, IEEE Trans. Software Engineering 5 (1979) 440, ACM Computer Survey 18, 1986

(Win86) Winograd T, Flores F, "*Understanding Computers and Cognition: A New Foundation for Design*", Ablex Publishers, Lexington, MA, 1986

(Grad87) Grady R B, Caswell D L, "Software Metrics: Establishing a Company-Wide Program", Prentice-Hall, 1987

(Lub87) Lubachevsky B D, "*Efficient Parallel Simulations of Asynchronous Cellular Arrays*", Complex Systems, vol.1, no.6, pp.1099—1123, S. Wolfram, (ed.), December, 1987

(CCIT) CCITT, "Specification and Design Language SDL", Blue Book, November 1988

(Holz88) Holzmann G J, "*An Improved Protocol Reachabily Analysis Technique*", Software Practice and Experience, vol. 18, no. 2, pp. 137-161, February 1988

(Litz88) Litzkow M, Livny M, Mutka M, "*Condor - a hunter of idle workstations*" in the Proceedings of the 8th International Conference of Distributed Computing Systems, IEEE Computer Society Press, San Jose, CA, January, 1988

(Ohno88) Ohno T, "*Toyota Production System: Beyond Large-Scale Production*", Productivity Press, 1988

(Ferg89) Ferguson D F, "*The Application of Microeconomics to the design of resource allocation and control algorithms in Distributed Systems*", PhD thesis, Columbia University, New York, 1989

(Leh89) Lehman M M, "*Uncertainty in Computer Application and its Control Through the Engineering of Software*", Journal of Software Maintenance: Research and Practice, v. 1, n. 1, Sept. 1989

(Blac90) Black A P, Artsy Y, "*Implementing location independent invocation*", Parallel and Distributed Systems, IEEE Transaction, Vol. 1, Issue 1, pp. 107 – 119, Littleton, MA; January, 1990

(Ferg90a) Ferguson D F, Nikolaou C Yemini Y, " *An Economy for Flow Control in Computer Networks*" in the Proceedings of the INFOCOM'90, 1990

(Ferg90b) Ferguson D F,Yemini Y, Nikolaou C, " *Microeconomic Algorithms for Load Balancing in Distributed Computer Systems*" in the Proceedings of the International Conference on Distributed Systems, ICDCS'90, 1990

(Levy90) Levy A M, Pavlides G, "*Simulation vs. prototype execution: a case study*", in Proceedings of the 1990 IEEE International Conference on Computer Systems and Software Engineering, pp. 428 – 436, May 1990

(Ram90) Ram R, Viswanadham N, "*Recent advances in queueing networks: a survey with applications to automated manufacturing*", ACE apos; Proceedings of XVI Annual Convention and Exhibition IEEE, pp. 89 – 93, India, 1990

(Shaw90) Shaw M, "*Prospects for an Engineering Discipline of Software*", IEEE Journal, Carnegie Mellon University, 1990

(Wat90) Waterman, Christopher A, Juju, "*A Social History and Ethnography of an African Music*", University of Chicago Press, 1990

(Yoji90) Akao Y, "*Quality Function Deployment: Integrating Customer Requirements into Product Design*" (Translated by Glenn H. Mazur), Productivity Press, 1990

(Holz91) Holzmann G J, "*Design and Validation of Computer Protocols*", Englewood Cliffs, N J, Prentice Hall, 1991

(Jeff91) Jeffrey J M, "*Using Petri nets to introduce operating system concepts*", Paper presented at the SIGCSE Technical Symposium on Computer Science Education, San Antonio, USA, 7-8 March 1991

(Jen91) Jensen K, Rosenberg G, "*High-level Petri nets: Theory and Application*", Springer-Verlag, 1991

(Holz92) Holzmann G J, Godefroid P, Pirottin D, "*Coverage Preserving Reduction Strategies for Reachabily Analysis*", in Proceedings for the IFIP/WG6.1 Symposium, Protocol Specification, Testing, and Verification (PSTV92), pp. 349-364, Orlando, Holland, June 1992

(IEEE1061) IEEE Standard 1061-1992, "*Standard for a Software Quality Metrics Methodology*", Institute of Electrical and Electronics Engineers, New York, 1992

(Rose92) Rose G B, "SSADM-the open methodology: Introduction to Software Design Methodologies", IEE Colloquium, pp. 6/1 - 6/5, Jun 1992

(Rose92) Rose G B, "*Introduction to Software Design Methodologies*", IEE Colloquium, June 1992

(Ferg93) Ferguson D F, Nikolaou C, Yemini Y " An *Economy for Managing Replicated Data in Autonomous Decentralized Systems*" in the Proceedings of the International Symposium on Autonomous Decentralized System, ISADS'93, Kawasaki, Japan, 1993

(Gar93) Garlan D, Shaw M, "*An Introduction to Software Architecture, in Advances in Software Engineering and Knowledge Engineering*" Vol 1, ed. Ambriola and Tortora World scientific Publishing Co., 1993

(Gib93) Gibbon E, "*The History of the Decline and Fall of the Roman Empire*", Everyman's Library, Oct. 1993

(Kha93) Khaddaj S.A., Haider N., Wilby M.R., et al, "*The application of graphics and animation techniques for large-scale simulations*", Transputer Applications and Systems 93, Vol: 36, pp. 244 – 259, 1993

(Paul93) Paulk M C, Weber C V, Garcia S M, Chrissis M B, Bush M W, "*Key Practices of the Capability Maturity Model, Version 1.1*", Software Engineering Institute, pp. 1, February 1993

(Stef93) Steffen B M, Sundrum E, "*Political Change opens Telecoms Markets*", Public Communication Networks, Siemens AG, Munich, 1993

(Ahr94) Ahrens G; Chandra A; Kanthanathan M; Cox D P, "*Evaluating HACMP/6000: a clustering solution for high availability distributed systems*", Fault-Tolerant Parallel and Distributed Systems,.in the Proceedings of IEEE Workshop, pp. 2 – 9, June, 1994

(Kazm94) Kazman R, Bass L, Abowd G, Webb M, "*SAAM: A Method for Analysing of Software Architectures*", in the Proceedings for the International Conference on Software Engineering, pp 81 – 90, May 1994

(Mal94) Malone T W, Crowston K, "*The interdisciplinary study of coordination*", ACM Computing Surveys (CSUR), 26(1), pp. 87–119, 1994

(Sing94) Singhal M, Shivaratri N G, "*Advanced Concepts in Operating Systems*", McGraw Hill, 1994

(Broo95) Brooks F P, "*The Mythical Man-Month*", Addison Wesley, 1995

(Hai95) Haider N, Khaddaj S A, Wilby M R, Vvedensky D D, "*Parallel Monte Carlo simulations of epitaxial growth*", Computers in Physics, v.9 n.1, pp.85-96, Jan- Feb,1995

(Leve95) Leveson N, "*Safeware: System Safety and Computers*", Addison-Wesley, 1995

(Mag95) Magee J, Dulay N, Eisenbach S, Kramer J, "*Specifying Distributed Software Architectures*", 5th European Software Engineering, ESEC, Sep.1995

(Tane95) Tanenbaum A S, "*Distributed Operating Systems*", Prentice Hall, 1995

(Wes95) Westbrook J, "*Load Balancing for Response Time*", in the Proc. of the 3rd Annual European Symposium on Algorithms, pp.355-368, September 25-27, 1995

(Brig96) Brightwell R, Skjellum A, "*MPICH on the T3D: A Case Study of High Performance Message Passing*", in Proceedings of the MPI Developers Conference, July 1996

(Ferg96) Ferguson D, Nikolaou C, Sairamesh J, Yemini Y, "*Economic models for allocating resources in computer systems. Market-Based Control: A Paradigm for Distributed Resource Allocation*", World Scientific Press, 1996, Singapore

(Ferg96) Ferguson D F, Nikolaou C, Sairamesh J, Yemini Y, "*Economic Models for Allocating Resources in Computer Systems*", ed. Scott Clearwater, 1996

(Henk96) Henke R, Konig A, Mitschele-Thie Al, Langendorfer P, "*Automated Derivation of Efficient Implementations from SDL Specifications*", Technical Report 11/96, IMMD VII, University of Erlangen-Nuremberg, 1996

(Jung96) Jung J II, "*Quality of Service in Telecommunicatios – Translation of QoS Parameters into ATM Performance Parameters in B - ISDN*", IEEE Communication Magazine, Vol. 34, pp. 112-117, August 1996

(Karl96) Karlsson J,"*Software Requirements Prioritizing*", in the Proceedings of the Second International Conference on Requirements Engineering (ICRE'96), IEEE Computer Society, pp. 110-116, Colorado Springs, CO, April 15-18, 1996

(Keen96) Keen C, "*Treatment of Metaphors in Software Engineering Education*", International Conference on Software Engineering: Education and Practice, SEEP, pp. 329, 1996

(Leue96) Leue s, Oechslin P, "*On Parallelising and Optimizing the Implementation of Communication Protocols*", IEEE/ACM Transactions on Networking, 4(1), 1996

(Lync96) Lynch N A, "*Distributed Algorithms*", Morgan Kaufmann, 1996

(Sim96) Simon H A, "*The Sciences of the Artificial*", MIT Press, 1996

(Kha96) Khaddaj, S. A., "*Stochastic Parallel Discrete Event Simulation Applied to Materials Design*", HPCS'96, IEEE Canada/Carlton University Press, pp. 391-399, Ottawa, Canada, 1996

(HOR97) Office of Inspector General, "*Changes in Operating Practices needed to improve the economy, efficiency, and effectiveness of House Telecommunications Functions*" Report No. 97, CAO-06, House of Representatives, Washigton, USA, March, 1997

(Berm97) Berman F, Wolski R, "*The AppLeS Project: A status report*", in the Proceedings of the 8th NEC Research Symposium, Berlin, Germany, May 1997

(Casa97) Casanova H, Dongarra J, "*NetSolve: A network server for solving computational science problems*", International Journal of Supercomputing Applications and High Performance Computing, pp. 11(3):212–223, 1997

(Chr97) Christensen S, Mortensen K H, "*Parametrisation of Coloured Petri Nets*", University of Aarhus, Computer Science Dept., Denmark 1997

(Gue97) Guerin R, Orda A, Williams D, "*QoS routing mechanisms and OSPF extensions*", in the Proceedings on Global Internet (Globecom), Phoenix, Arizona, November 1997

(Henk97) Henke R, Konig, Mitschele-Thiel A, "*Derivation of Efficient Implementations from SDL Specifications Employing Data Referencing, Integrated Packet Framing and Activity Threads*", SDL' 97, Proceedings of the 8th SDL-Forum, 1997

(Jen97) Jensen K, "*Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use*", Monographs in Theoretical Computer Science, Springer-Verlag, Vol 2, 1997

(UML97) UML, "UML Notation Guide version 1.1", Rational Software Corporation, September 1997

(Yam97) Bar-Yam Y, "*The Dynamics of Complex Systems (Studies in Nonlinearity)*", August, 1997

(Bol98) Bolcer G A, Taylor R N, "*Advanced workflow management technologies*", Software Process Improvement and Practice, vol 4, no. 3, 125-171, 1998

(Cave98) Cavendish D, Gerla M, "*Internet QoS Routing using the Bellman-Ford Algorithm*", in IFIP Conference on High Performance Networking, 1998

(Gross98) Gross D; Harris CM, "*Fundamentals of Queueing Theory*" 3rd Edition,Wiley series in Probability and Statistics, Wiley, 1998

(Krem98) Kremer R, "*Formal specification - what is a formal specification – Requirement Engineering*", SENG 611 Requirements Engineering, 1998

(Kris98) Kristensen L M, Christensen S, Jensen K, "*The practitioner's guide to Coloured Petri Nets*", International Journal on Software Tools for Technology Transfer, pp. 98–132, 1998

(Maha98) Mahadevan I, Sivalingam KM, "*An Architecture for QoS guarantees and Routing in Wireless/Mobile Networks*", pp.11-20, ACM WOWMOM'98, Dallas, Texas, USA, 1998

(Oud98) Oudrhiri R, Desreumaux M, "*Information and Software Systems: from Architecture to Urbanism*", In the Proceedings of the WISCAI Conference, IFIP, Chapman & Hall, 1998

(Ullm98) Ullman J D, "*Elements of ML Programming*", Prentice - Hall, 1998

(Arn99) Arnold K, Scheifler R, Waldo J, O'Sullivan B, Wollrath A, "*Jini Specification*", Addison-Wesley Longman Publishing Co., 1999

(Asch99) Ascher D, Lutz M, "*Learning Python*", O'Reilly & Associates, March 1999

(Chap99) Chapin S, Karpovich J, Grimshaw A, "*The Legion resource management system*" in the Proceedings of the 5th Workshop on Job Scheduling Strategies for Parallel Processing, Springer – Berlin, San Juan, Puerto Rico, April, 1999

(Cost99a) Cost R, Chen Y, et al, "*Modeling agent conversations with colored petri nets*", May 1999.

(Cost99b) Cost R, Chen Y, et al "*Using colored petri nets for conversation modeling*", 1999

(Estu99) Estublier J, "Is a process formalism an Architecture Description Language?", Dassault Système / LSR, International Process Technology Workshop (IPTW), Villard de Lans, France, September, 1999

(Grea99) Greaves M, Holmback H, Bradshaw J M, "*Agent conversation policies*", Handbook of Agent Technology. Cambridge, MA: AAAI Press / MIT Press. Forthcoming, 1999

(Gutt99) Guttman E, Perkins C, Veizades J, Day M, "*Service Location Protocol, Version 2*", IETF, 1999

(Haa99) Haas Z J, Liang B, "*Ad-Hoc Mobility Management with Randomized Database Groups*", in Proceedings of the IEEE International Conference on Communication, IEEE Computer Society, pp. 1756-1762, 1999

(Ho99) Ho W M, Jézéquel J M, Guennec M, Pennaneac'h F, "*UMLAUT: an extendible UML transformation framework*", In Proceedings of the Automated Software Engineering, ASE'99, Florida, October 1999

(Holm99) Holmback H, Greaves M, and Bradshaw J M, "*A pragmatic principle for agent Communication*", Bradshaw J M, Etzioni O, Mueller J (ed.), in the Proceedings of Autonomous Agents, ACM Press, Forthcoming, New York, 1999

(Kapa99) Kapadia N, Fortes J, "*PUNCH: An architecture for Web-enabled wide-area network-computing. Cluster Computing*", the Journal of Networks, Software Tools and Applications, pp. 2(2):153–164, 1999

(Liu99) Liu B, Guo Y, Kurose J, Towsley D, Gong W, "*Fluid simulation of large scale networks: issues and tradeoffs*", in the Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications, June 28-July 1, Vol. 4, pp. 2136-2142, Las Vegas, 1999

(Mic99) Microsoft Corporation, "*Universal Plug and Play: Background*", Microsoft Corporation, 1999

(Nic99) Nicol D, Goldsby M, Johnson M, "*Fluid-based Simulation of Communication Networks using SSF*", in the Proceedings of the 1999 European Simulation Symposium, October 1999, Erlangen-Nuremberg, Germany

(Pax99) Paxson V, "*End-to-End Internet Packet Dynamics*", IEEE/ACM Transactions on Networking 7 (3), pp. 277-292, 1999

(Sun99) Sun W, "QoS, *Policy, Constraint Based Routing*", Computer Science and Engineering, Ohio State University, 2002

(Jaco99) Jacobs R, "*Evaluating customer satisfaction with media products and services: an attribute based approach*", European Media Management Review, 1999

(Buyy00a) Buyya R, Abramson D, Giddy J, "*Nimrod/G, An architecture for a resource management and scheduling system in a global computational Grid*", in the Proceedings 4th International Conference and Exhibition on High Performance Computing in Asia - Pacific Region, IEEE Computer Society Press, Beijing, China, May, 2000

(Buyy00b) Buyya R, Giddy J, Abramson D, "*An evaluation of economy-based resource trading and scheduling on computational power Grids for parameter sweep applications*", in the Proceedings of the 2nd International Workshop on Active Middleware Services, Kluwer Academic Press, Pittsburgh, August 2000

(Dub00) Dubrovsky A, Gerla M, Lee SS, Cavendish D, "*Internet QoS Routing with IP Telephony and TCP Traffic*", in the Proceedings of the ICC, June 2000

(Gall00) Gallouj F, "Innovation *in the Service Economy: The New Wealth of Nations*", Edward Elgar Publishing, 2002

(Grea00) Greaves M, Holmback H, Bradshaw J, "*What is conversation Policy*", Mathematics and Computing Technology, The Boeing Company, 2000

(Herm00) Hermann R, Husemann D, Moser M, Nidd M, Rohner C, Schade A, "*DEAPspace: Transient Ad-Hoc Networking of Pervasive Devices*", in Proc.of the 1st ACM International Symposium on Mobile Ad Hoc Networking & Computing, IEEE Press, pp. 133-134, Boston, Massachusetts, USA, 2000

(Li00) Li J, Jannotti J, Couto D S J D, Karger D R, Morris R, "*A Scalable Location Service for Geographic Ad Hoc Routing*", in Proceedings of the 6th Annual International Conference on Mobile Computing and Networking, ACM Press, pp. 120-130, Boston, Massachusetts, USA, 2000

(Lilu00) Lilius J, Paltor I V, "*vUML: a Tool for Verifying UML Models*", TUCS, Technical Report, No. 272, May 1999, August 2000

(Lub00) Lubachevsky B D, "*Fast Simulation of Multicomponent Dynamic Systems*", Bell Labs Technical Journal, Vol.5, No.2, pp.134-156, April-June 2000

(Mikk00) Mikkonen T; Lahde E; Niemi J; Siiskonen M, "*Managing software evolution with the service concept*", in the Proceedings of International Symposium on Principles of Software Evolution, pp 46 – 50, 2000

(Morr00) Morrison R, Balasubramaniam D, Greenwood R M, Kirby G N C, Mayes K, Munro D S, Warboys B C, "*A Compliant Persistent Architecture, in Software Practice and Experience*" vol. 30, no. 4, pp. 363 – 386, 2000

(Star00) START, "*Selected Topics in Assurance Related Technologies*", Reliability Analysis Centre, Vol. 6, No. 5, 2000

(Sub00) Subramanyan R, Miguel-Alonso J, Fortes J A B, "*A scalable SNMP-based distributed monitoring system for heterogeneous network computing*", High Performance Networking and Computing, in the Proc.of the ACM/IEEE conf. on Supercomputing, Art. 14, Dallas, Texas, USA, 2000

(Int00) Intven, Hank, "*Telecommunications Regulation Handbook*", World Bank, infoDev Program, Washington, D.C., 2000

(Buyy01) Buyya R, Abramson D, Giddy J, "*A case for economy Grid architecture for service-oriented Grid computing*", in the Proceedings of the International Parallel and Distributed Processing Symposium: 10th IEEE International Heterogeneous Computing Workshop (HCW 2001), IEEE Computer Society Press, San Francisco, CA, April 2001

(Fos01) I. Foster, C. Kesselman, and S. Tuecke, "*The Anatomy of the Grid: Enabling Scalable Virtual Organizations*" Int'l Journal High-Performance Computing Applications, vol. 15, no. 3, 2001

(Gros01) Grosso W, "*Java RMI*", O'Reilly 1st Edition, October 2001

(Ham01) Hamdouch A, Samuelides E, "*Innovation's Dynamics in Mobile Phone Services in France*", Maison des Sciences Economiques, Université de Paris, I-Panthéon, Sorbonne, France, 2001

(IBM01) IBM Everyplace Wireless Gateway, "*Short message support in the Websphere Everyplace Suite*", IBM Corporation, Research Triangle Park, 2001

(Jenn01) Jennings N R, "*An Agent-based approach for building complex software systems*", Communications of the ACM, Vol 44, No. 4, April 2001

(Jini01) Sun Microsystems, "*Jini Network Technology - An Executive Overview*", Sun Microsystems Inc, Palo Alto, USA, 2001

(Kou01) Koutsoukos G, Gouveia J, Andrade L, Fiadeiro J L, "*Managing Evolution in Telecommunications Systems*", IFIP Conference Proceedings, Vol. 198, 2001

(Kout01) Koutsoukos G, Gouveia J, Andrade L, Fiadeiro J L, "*Managing Evolution in Telecommunication Systems*", In the Proceedings of the IFIP TC6 / WG6.1 3rd International Working Conference on New Developments in Distributed Applications and Interoperable Systems; Vol. 198, 2001

(Lore01) Lorentsen L, Tuovinen AP, Xu J, "*Modelling Feature Interaction Patterns in Nokia Mobile Phones using Coloured Petri Nets and Design/CPN*", In Jensen, ed., Proceedings of the third workshop on practical use of Coloured Petri Nets and the CPN Tools, August 2001

(Lub01) Lubachevsky B D, Weiss A, "*Synchronous Relaxation for Parallelising Spin Simulations*", 5th Workshop on Parallel and Distributed Simulation, pp. 185-192, California, May 2001

(Rose01) Rosen E, Viswanathan A, Callon R, "*Multiprotocol label switching architecture*", Request for Comments 3031, Internet Engineering Task Force, January 2001

(Shaw01) Shaw M, "*The coming-of-age of software architecture research*", in Proceedings of ICSE, pp. 656–664, Carnegie Mellon University, 2001

(Somm01) Sommerville I, "*Software Engineering*", vol. 6, Pearson & Addison Wesley, 2001

(Ster01) Stergiou C, Arys G, "*Policy based agent management using conversation patterns*", in Proceedings. Of 5th International Conference on Autonomous Agents, pp. 220 – 221, Quebec, Canada, 2001

(Xue01) Xue Y, Li B, Nahrstedt K, "*A Scalable Location Management Scheme in Mobile Ad-Hoc Networks*", in Proceedings of the 26th Annual IEEE Conference on Local Computer Networks: IEEE Computer Society, pp. 102-112, 2001

(Zhan01) Zhang T, "*Agent-based Interperability in Telecommunications Applications*", PhD Thesis, University of Berlin, 2001

(AFQ01) Air Force Logistics, "*Quotes for the Air Force Logistician*", Air Force Logistics Management Agency, Alabama, USA 2001

(Teo01) Teo Y M, "*Comparison of Load Balancing Strategies on Cluster-based Web Servers*", SIMULATION, Simulation Council Inc., Vol 77, no. 5-6, pp. 185-195, Singapore, 2001

(Page01) Page M, "*Internetworking Technologies Handbook*", Cisco Press, Cisco Systems Inc., September 2001

(Awer02) Awerbuch B, Holmer D, NitaRotaru C, Rubens H, "*An On Demand Secure Routing Protocol Resilient to Byzantine Failures*", WiSe'02, Atlanta, Georgia, US, September 28, 2002

(Buyy02) Buyya R, Abramson D, Giddy J, Stockinger H, "*Economic models for resource management and scheduling in Grid computing*", Concurrency and Computation, John Wiley & Sons, January 2002

(Chak02) Chakraborty D, Joshi A, Finin T, Yesha Y, "*GSD: A Novel Group Based Service Discovery Protocol for MANETS*", in Proceedings of the 4th IEEE Conference on Mobile and Wireless Communications Networks,MWCN, IEEE Press, Stockholm, Sweden, 2002

(Chan02) Chance B L, "Components *of Statistical Thinking and Implications for Instruction and Assessment*", Journal of Statistics Education,vol 10, no.3, 2002

(Cho02) Cho, Sung Woon, "The *Telecom-Economy Nexus: Innis and Du Boff Revisited*", Delft University Press, 2002

(Chow02) Chowdhury S, "*The Power of Design for Six Sigma*", Dearborn Trade, 2002

(Fos02) Foster I, Kesselman C, Nick J. M, Teucke S, "*Grid Services for Distributed System Integration*", IEEE Publication, 2002

(Fut02) Futral W T, "*InfiniBand Architecture, Development and Deployment*", A Strategic Guide to Server I/O Solutions", Intel Press, January 2002

(Hel02) Helal S, Desai N, Verma V, Lee C, "*Konark: A Service Discovery and Delivery Protocol for Ad-hoc Networks*", in Proceedings of the 3rd IEEE Conference on Wireless Communication Networks WCNC, IEEE Press, New Orleans, USA:, 2002

(Kor02) Korniss G, Novotny M A, Kolakowska A K, Guclu H, "*Statistical properties of the simulated time horizon in conservative parallel discrete-event simulations*", Proceedings of the ACM symposium on Applied computing, Madrid, Spain 2002

(Lau02) Lauesen S, "*Software Requirements - Styles and techniques*", Addison Wesley, 2002

(Lync02) Lynch N, Shvartsman A, "*Communication and Data Sharing for Dynamic Distributed Systems*", pp. 62–67, 2003, Springer-Verlag, Heidelberg, Berlin 2003

(Mas02) Kawashima M, "*Telecom Value Chain Dynamics and Carriers' Strategies in Converged Networks*" Management of Technology, MIT, June 2002

(Med02) Medhi D, "*Quality of Service (QoS) Routing Computation with Path Caching: A Framwork and Network Performance*", IEEE Communication Magazine, December 2002

(Mull85) Mullender S J, Vitanyi P M B, "*Distributed Match-Making for Processes in Computer Networks (preliminary version)*", in Proceedings of the 4th Annual ACM Symposium on Principles of Distributed Computing., ACM Press, pp. 261-271,Ontario, Canada:, 1985

(Nam02) Namsoo J, Lancaster B, "*In Search of Differentiation*", LTC International In, April 2002

(Oud02) Oudrhiri R, "*Une approche de l'évolution des systèmes,- application aux systèmes d'information*", ed.Vuibert, 2002

(Sist02) Sistare S J, Jackson C J, "*Ultra-High Performance Communication with MPI and the Sun Fire Link Interconnect*", In Proceedings of Supercomputing, 2002

(Tan02) Tanaka K, MeiHong Zeng, Kashimori Y, Kambara T, "*Evolution of human communication method from vocal signals associated with gesture to syntatic language*", in Proceedings of the 9th International Conference onVolume 4, Issue, pp. 1941 – 1945, Nov 2002

(VMR02a) Empower Interactive R&D, "*Virtual Mobile Redirector Manual Specification Guide*", Empower Interactive Group Ltd, 2002

(VMR02b) Empower Interactive R&D, "Virtual *Mobile Redirector Object Oriented Model Specification Guide*", Empower Interactive Group Ltd, 2002

(Yuan02) Yaun J, Mills K, "*Exploring Collective Dynamics in Communication Networks*", Journal of Research of the National Institute of Standards and Technology, vol.107, no. 2, March–April 2002

(Zhu02) Zhu F, Mutka M, Ni L, "*Classification of Service Discovery in Pervasive Computing Environments*", MSU-CSE-02-24, Michigan State University, 2002

(Ada02) Adan I, Resing J, "*Queueing Theory*", Department of Mathematics and Computing Science, Eindhoven University of Technology, Netherlands, 2002

(Alm03) Almeida J P; van Sinderen M; Pires L F; Quartel D, "*A systematic approach to platform-independent design based on the service concept*", In the Proceedings of the Enterprise Distributed Object Computing, 7th IEEE International Conference, pp. 112 – 123, September, 2003

(Boe03) Boesgaard C, "*Anonymous communication in practice*" Department of Computer Science, University of Copenhagen, Denmark, 2003

(Bre03) Breyfogle III F W, "*Implementing Six Sigma Smarter Solutions Using Statistical Methods*", 2nd Edition, Wiley, 2003

(Chu03) Chulani S Ray, B Santhanam P, Leszkowicz R, "*Metrics for managing customer view of software quality*", in Proceedings of 9th International Software Metrics Symposium, pp. 189 – 198, 3rd – 5th September 2003

(Crev03) Creveling C M, Slutsky JL, Antis D Jr, "*Design for Six Sigma in Technology and Product Development*", Pearson Education Inc, Prentice Hall, 2003

(Dede03) Dedecker J, De Meuter W, "*Communication abstractions through new language concepts*", July, 2003

(Gao03) Gao X; Wu G, Miki T, "*QoS framework for mobile heterogeneous networks*", ICC apos; 03, IEEE International Conference on Communications, pp. 933 – 937, May 2003

(Koz03) Kozat U C, Tassiulas L, "*Network Layer Support for Service Discovery in Mobile Ad Hoc Networks*", in Proceedings of IEEE INFOCOM, vol. 23, IEEE Computer Society, pp. 1965-1975, 2003

(Nem03) Nemeth E, Lakner R, Hangos M, Cameron I T,"Hierarchical CPN model-based diagnosis using HAZOP knowledge", Research Report SCL -009, 2003

(Nik03) Nikolic N, "*Limitations of theoretical and commonly used simulation approaches inconsidering Military Queuing Systems*", in Proceedings of 15th European Simulation Symposium, Ed Vlatka Hlupic, SCS European Council, 2003

(Novo03) Novotny M A, Kołakowska A K., Korniss G, "*Algorithms for faster and larger dynamic Metropolis simulations*" in Proceedings of AIP Conference Vol. 690, pp. 241 2003

(OFT03) OFTA, "*Quality of Service (QoS) in the Telecommunications Industry*", Discussion Paper, Office of Telecommunications Authority, August 2003

(Oud03) Oudrhiri R, Desreumaux M, Préfol P, Bernier A, "*Inductive integration of data and processes for assisting closure of nuclear power plant*", Software Architecture: Interweaving Information Systems, ICSSEA, Electricite de France EDF, IISCA, Paris 2003

(Oxf03) Oxford University, "Oxford Dictionary of ENGLISH", Oxford University Press, 2Rev Ed edition, August 2003

(UKWS03) Wireless World Forum, "*UK Wireless Statistics: Analyst Statistics Report*", May 2003

(Yang03) Yang & El Haik, "*Design for Six Sigma*", McGraw-Hill, 2003

(Debu03) Debusmann M, Keller A, "*SLA-driven Management of Distributed Systems using the Common Information Model*", University of Applied Sciences, Kurt-Schumacher-Ring 18, IBM Research Division, 2003

(Chan03) Chandra S, Parashar M, Hariri S, "*GridARM: An Autonomic Runtime Management Framework for SAMR Applications in Grid Environments*", Autonomic Applications Workshop, *High Performance Computing* (HiPC'03), pp. 286-295, India, December 2003

(Cis04) Cisco Systems, "*Gateway Load Balancing Protocol Overview*", Cisco Systems Inc., 2004

(Dam04) Damsgaard J, Gao P, "*Mobile telecommunications market innovation: The transformation from 2g to 3g*", Copenhagen Business School, Denmark 2004

(Fung04) Hay Fung K, Low G, Bay P K, "*Embracing dynamic evolution in distributed systems Software*", IEEE Vol. 21, Issue 2, pp. 49 – 55, March-April, 2004

(Gou04) Gouaich A, "*Coordination and Conversation Protocols in Open Multi-agent Systems*", Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2004

(Kha04) Khaddaj S A, Horgan G, "*The Evaluation of Software Quality Factors in Very Large Information Systems*", School of Computing and Information Systems, Kingston University, UK, Nexis Associates Ltd, London, 2004

(Kim04) Kim D H, B Kahng, Kim D, "*Multi-component static model for social networks*", The European Physical Journal, Condensed Matter and Complex Systems, Springer Berlin / Heidelberg, Vol. 38, No. 2, March 2004

(Sriv04) Srivastava S, et al, "*Benefits of Traffic Engineering using QoS Routing Schemes and Network Controls*", Computer Communications Vol. 27, no. 5, pp. 387-399, March 2004

(Vila04) Vilas J F, Arias J P, Vilas A F, "*High Availability with Clusters of Web Service*", Advanced Web Technologies and Applications, ed.Springer Berlin / Heidelberg, Vol. 3007/2004, March, 2005

(Will04) Williams L G, Smith C U, "Web Application Scalability: A Model-Based Approach", Software Engineering Research and Performance Engineering Services, 2004

(Zeu04) Zeus Technology, "*Traffic Manager Evolution*", Zeus Technology Press, Cambridge, England, 2004

(Mak04) Makoond B, Khaddaj S, Saltmarsh C, "*Conversational Dynamics*", Empower Interactive Press, EIGroup, February 2004

(Buyy04) Buyya R, *GridBus: "A Economy-based Grid Resource Broker", University of Melbourne, 2004*

(In04) In I J, Avery P, Cavanaugh R, Ranka S, "*Policy Based Scheduling for Simple Quality of Service in Grid Computing*", in the International Parallel & Distributed Processing Symposium (IPDPS), Santa Fe, New Mexico, April 2004

(Bersini05) Bersini H, "*Des réseaux et des sciences – Biologie, informatique, sociologie: l'omniprésence des reséaux*", Vuibert, Paris 2005

(Cho05) Cho C, Lee D, "*Survey of Service Discovery Architectures for Mobile Ad hoc Networks*", Mobile Computing, CEN 5531, Department of Computer and Information Science and Engineering, CICE, University of Florida, 2005

(Enge05) Engelstad P E, Zheng Y, "*Evaluation of Service Discovery Architectures for Mobile Ad Hoc Networks,* in Proceedings of the 2nd Annual Conference on Wireless On-demand Network Systems and Services, WONS, 2005

(Grub05) Gruber H, "*The Economics of Mobile Telecommunications*", Cambridge University Press, 2005

(GSM05) GSMA Analysis Report, "*Growth of the global digital mobile market*", GSM Association, Feb 2005

(Hay05) Hayler R, "*What is Six Sigma Process Management*", McGraw Hill, 2005

(Kha05) Khaddaj S, Horgan G, "*A Proposed Adaptable Quality Model for Software Quality Assurance*", School of Computing and Information Systems, Kingston University, Kingston upon Thames, 2005

(Maui05) Cluster Resources, "*Maui Scheduler Administrator Guide*", version 3.2, 2005

(Mel05) Melody W, Sutherland E, Tadayoni R, "*Convergence, IP Telephony and Telecom Regulation*" Workshop on Convergence, VoIP and Regulation, Telecommunication Regulatory Authority of India (TRAI), New Delhi, Mar 2005

(Mer05) Merriam-Webster, "*The Merriam-Webster Dictionary*", Merriam-Webster, New Edition, Jul. 2005.

(Miya05) Miyazakia K, Wiggersb E, "*Innovation in Telecom Services Framework and Analysis Based on the Case of International Pre-paid Calling Cards in Japan*", Tokyo Institute of Technology, School of Information Management, Tokyo Japan, 2005

(Ross05) Rossi D, Turrini E, "*Analyzing the Impact of Components Replication in High Available J2EE Clusters*", Joint International Conference on Autonomic and Autonomous Systems and International Conference on Networking and Services, pp. 56, October, 2005

(Tom05) Tomkins W, "*Smoke Signals*", The Inquiry Net, Feb. 2005

(Clur05) Cluster Resources, "*Maui Scheduler Administrator Guide*", version 3.2, Cluster Resources Press, Cluster Resources Inc., 2005

(Pres05) Pressman R S, "*Software Engineering : A Practitioner's Approach*", R S Pressman & Associates Inc Publishing, 6th Edition, 2005

(Volt05) Voltaire, "*Voltaire® ISR 9096 InfiniBand Switch Router*", Voltaire Inc, 2004

(Mell05) Mellanox Technologies, "*InfiniHost™ III Ex Dual-Port InfiniBand HCA Cards with PCI Express x8 Technical Spécifications*", Mellanox, USA, 2005

(Bai06) Bai X, Bölöni L, Marinescu D C, Siegel H J,Daley R A, Wang I, "*A brokering framework for large-scale heterogeneous systems*", 15th Heterogeneous Computing Workshop (HCW 2006) in the Proc. of IPDPS 2006, Rhodes, 2006

(Her06) Herdan B, "*The Customer Voice in Transforming Public Services*", Independent Report from Review of the Charter Mark Scheme, Cabinet Office UK, Jun 2006

(Int06) Intel, "*Trend in Telecom: The Next Big Idea in Service Oriented Networks*", Intel Corporation, 2006

(MDA06) Mobile Data Association, "*SMS Statistics Report*", MDA Statistics Release 2006

(Oud06) Oudrhiri R, "*Implementing the DFSS culture in an IT and Software Environment*", WBCF, San Francisco, 2006

(PNT06) PNTDB, "*Petri Nets Tools Database*", retrieved October 23, 2006

(Slee06) Sleeper A, "*Design for Six Sigma Statistics: Six Sigma Operational Methods*", McGraw-Hill, 2006

(SMS06) mCube, "*SMS Gateway API reference*", version 2.0, mCube Studio, February 17th, 2006

(B-IP06) Big-IP TLM, "*F5 Security Solutions for Messaging Systems*", F5 Networks Press, September 06

(Cis06a) Cisco Systems Case Study, "*U.S. Telco Increases the Value of its Siebel Enterprise Applications*", Cisco Systems Public Information, Cisco System Inc.,2006

(Cis06b) Cisco Systems Data Sheet, "*Cisco CSS 11500 Series - Content Services Switch*", Cisco Systems Inc., 2006

(IEEEC06) IEEE Computer Society Task Force on Cluster Computing

(Szy06) Szyperski C, "*Component-Based Software Engineering*", in the Proc. Of the 9th Int. Symposium, CBSE'06, Vol. 4063, Västeras, Sweden, June 29 - July 1, 2006

(Ong06) Ong D, Sileika R, Khaddaj S, Oudrhiri R, "*Data Management System Evaluation for Mobile Messaging Services*", ICEIS, pp. 226-231, 2006

(Gao07) Gao P, Damsgaard J, "*A framework for understanding mobile telecommunications market innovation: a case of china*", IDPM, University of Manchester, UK 2007

(NSN07) Nokia Siemens Networks, "*IP - Centric Convergence*", Industry Themes, Nokia Siemens publications, 2007

(SDG07) Screen Digest, "*The mobile content revolution: How gaming, music and TV will transform the mobile market by 2011*", Screen Digest Global Media Intelligence, May 2007

(Del07) Dell'Oro Market Research Services, "*The Dell'Oro Group Ethernet Switch Quarterly Report*", Networking and Telecommunications Report, Dell'Oro Group Press, August 22, 2007

(VMW07) VMWare, "*Round-Robin Load Balancing*", VMWare Technical Notes, VMWare Inc., Palo Alto, CA., 2007

(Cis07) Cisco Systems, "*Understanding CSM Load Balancing Algorithms*", Cisco Technical Document, Cisco Systems Inc., May 16, 2007

(Park07) Park H M, "*Comparing Group Means: The T-test and One-way ANOVA Using Stata, SAS, and SPSS*", Indiana University, 2007

(Mak07) Makoond B, Khaddaj S, Ong D, Oudrhiri R,"*Novel Distributed Computing Techniques for Mobile Telecommunications*", in the 6th International Conference on Distributed Computing and Applications for Business, Engineering and Science, August, 2007

PAGE/PAGES EXCLUDED UNDER INSTRUCTION FROM THE UNIVERSITY