Automated, Ubiquitous Delivery of Generalised Services in an Open Market

Leslie J. Green

A Dissertation Submitted for the Degree of
Doctor of Philosophy in Computing Science
at the
University of Technology, Sydney

Faculty of Information Technology
University of Technology, Sydney

Sydney, Australia

2007

Certificate of Authorship/Originality

I certify that the work in this thesis has not previously been submitted for a degree nor has it been submitted as part of requirements for a degree except as fully acknowledged within the text.

I also certify that the thesis has been written by me. Any help that I have received in my research work and the preparation of the thesis itself has been acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

INTENTIONALLY BLANK

To my wife, for everything...

Acknowledgements

My doctoral research culminating in the following dissertation was a long and involved process. There are many people to thank for their help and support during this time:

First and foremost to my supervisor John Debenham for his energy and attitude, maintaining an enjoyable research process throughout my candidature. John helped greatly with my technical writing and clarification of ideas, essential to this dissertation. The comments provided on multiple chapter drafts were most helpful!

To my wife, Iris for the motivation and emotional support throughout the more difficult times. Along with my family and friends, your help to keep me grounded in the real world while my mind was elsewhere was invaluable.

Many thanks to all my colleagues from the Alcatel Collaboration, both in Sydney and France. The ability to base my research on a real industry problem, and receive guidance and feedback was essential.

To my co-supervisor Simeon Simoff, fellow red-square denizens and colleagues within the E-Markets research group, especially Paul Bogg, whom I consulted on numerous occasions. I thank you all for the help you have provided along the way.

A big thankyou to my good friend Denn, who helped me by proof reading this document.

Finally, to the Institute of Information and Communication Technologies (IICT), Alcatel and the Australian Research Council. Without your generous financial support my candidature would have been far more difficult, if not impossible.

This work has been performed as part of an Australian Research Council Linkage Grant, LP0560935 between Alcatel and the University of Technology, Sydney. The project title is: Managing **Q**uality of Experience **D**elivery **In** **N**ew Generation Telecommunication Networks with **E**-negotiation **(QDINE)**.

Abstract

Telecommunications networks, and the services delivered over those networks have become an integral part of most people's lives in the developed world. The range and availability of these services is increasing, however the management of services still lags well behind technical capability, providing unnecessary barriers to the adoption of available technology. The work described in this dissertation has a primary goal of enabling flexible, automated delivery of any telecommunication-based service. More specifically, a mechanism to solve the administrative problems in enabling end users to automatically establish service agreements for any available service, from any available provider.

The aims of this work are to:

1. enable the description of service level agreements(SLA) for generalised telecommunication-based services, and
2. provide mechanisms by which those service level agreements may be managed.

The term "generalised services" means that all service types are managed using a common framework and set of processes.

To derive at a suitable service level agreement description language, the characteristics of telecommunication-based services are first analysed, along with considerations in delivering a service, including service quality, resource allocation and configuration, service pricing and service ubiquity. The current art in SLA description is studied and the requirements of an appropriate language are proposed. An ontological approach to SLA description is adopted, and an SLA description language is developed based on semantic web technologies.

To develop the mechanisms for SLA management, the current art is first analysed, and a set of requirements for a suitable SLA management framework are proposed. These requirements are used to guide the design of a multi-agent SLA negotiation framework, including a detailed description of the communication model, framework processes, and social behaviour of the agents involved.

Finally, the SLA description language and the negotiation framework are compared with the closest art, and are assessed against tightly argued criteria. An experimental framework and use cases are developed to explore an application of the proposed solution, and to validate completeness.

The approach taken has led to the following two key contributions:

1. A set of formal ontologies that may be used to semantically describe secure service level agreements for any application domain.
2. A multi-agent system providing an open market where services can be discovered, participants identified, and negotiation performed using context specific mechanisms.

The conclusions of the work are that an ontology-based SLA description language is appropriate for describing generalised SLAs, and that a distributed, agent based negotiation platform that is based on an open market and uses a minimal set of core processes with an extensible, ontology based communication mechanism is appropriate for managing service level agreements in a generalised, automated and ubiquitous way.

Table of Contents

List of Figures

List of Tables

List of Symbols

α - Agent acting in Market Agent role

β - Agent acting in Service Provider role

γ - Agent acting in Consumer role

δ - Agent acting in Billing Provider role

ϵ - Agent acting in Subscriber role

ζ - A QDINE agent

κ - A service

λ - Message content language

μ - FIPA message payload

π - Agent state in a protocol

ρ - Interaction Protocol

ϱ - An agent role

σ - Message Reply identifier

τ - Message Reply-By time

ϕ - Performative

φ - Root message content individual

χ - Message payload content

ψ - Conversation

ω - Ontology

A - Set of all agents acting in the Market Agent role

B - Set of all agents acting in the Service Provider role

Γ - Set of all agents acting in the Consumer role

E - Set of all agents acting in the Subscriber role

Z - Set of all QDINE agents

Chapter 1
Introduction

Telecommunications is at the core of what some describe as the current "information age"[1]. Service delivery over telecommunication networks is increasing, and the complexity of managing these services is also increasing. Motivation for this work came from two key problems in current telecommunications:

1. Telecommunication-based services with a guaranteed quality are currently not available without the use of some specialised, manual process to establish the service subscription, typically at great cost and effort. Common place realisation of quality-assured new generation services such as remote medical diagnosis, online learning or even cheap, reliable video conferencing is not available with current technology.
2. Telecommunication networks are everywhere, however only a small fraction of them are typically available to a user. A user may be forced to use a busy, expensive network to carry a service while there is a cheap, unused network also present, purely because the user has no usage agreement with the unused network. It is desirable to use whichever network is most attractive at a given time to perform a desired task.

Service management must be improved to achieve a smooth delivery of services over a telecommunications infrastructure.

1.1 A Closer Look

New Generation Networks attempt to offer additional benefits to both consumers and providers in the telecommunication value chain. They are designed to enhance end-user Quality of Experience by providing greater choice of services, breaking the traditional fixed service offerings and allowing the user to do what they want, when they want, where they want at a given price. Service Providers stand to benefit from these customised service offerings through the ability to gain additional income from the value-added services.

A current trend in service delivery is a movement towards wireless networks. This is largely due to two factors:

- First, consumers desire the convenience of mobile connectivity. Freedom from the confines of wires is sought and consumers are prepared to pay a reasonable price for that luxury. That price may be a lower quality, higher risk of disconnection or simply higher monetary cost [2]. If these factors can be maintained, a user's Quality of Experience may be vastly improved.
- Second, providers seek to minimise the large capital and operational expense in construction and maintenance of a wired local loop. This is especially evident in countries with insufficient existing fixed infrastructure. For example, Latin America is the one of the largest current broadband wireless markets in relation to population with significant pre-WiMAX deployment [3].

Recently, there has been increased effort towards ubiquity in wireless services such as in [4] or [5]. The goal of ubiquity is the convergence of heterogeneous networks from both a technological and administrative point of view. Within ubiquitous service delivery, an end user is free to roam within the constraints of any possible network connectivity. Consequently, services utilised by end users may be delivered via numerous service domains. Providers of these domains will want to be reimbursed for the provision of a service, and consumers may desire some certainty guarantees on the service delivered. A method for service management in a ubiquitous environment is required.

1.2 The Desired Scenario

John, a relatively tech-savvy individual, lives in Sydney and enjoys an array of devices used for work, communication, entertainment and education. His devices include a handheld GPS/Telephone/Camera/PDA, a laptop computer and home media centre. From any device, a list of telecommunication-based services is available, which dynamically changes according to service/provider availability and personal preferences.

It is 2pm on Sunday and John has a flight booked to see his mother in London, departing at 6pm that day. Sitting on his lounge, John checks his flights on the laptop to see if they are still on-time. Following, he calls his mother using a video call from the media centre. John is a poor university student, but still likes to use decent quality calls, so he asks to be alerted when the price of a medium quality video call falls below 50c/minute. In the meantime, John chooses to watch a movie from one of the many Video on Demand providers available. John eventually receives an alert about his video call and calls his mother.

At 3pm John leaves for the airport by train while finishing the movie using his PDA. On the way, John receives a call from a client of his part-time help desk job. With a 30 minute train ride remaining, John uses his laptop to remotely accesses the client's network to make some changes. Throughout the train journey, network access is provided by GSM, UMTS, Satellite and WiMax networks as they come in and out of availability. Network services are automatically selected by John's computer according to the requirements of the remote access application.

On arriving at the airport, John checks-in his bags and heads to the local coffee shop to recharge and finish off his work using the available WiFi. John calls his girlfriend to say goodbye via his handheld device, which also uses the WiFi network while it's available and of sufficient quality.

1.3 The Problem – Behind the Scenes

The key word is ***Flexibility***. Current telecommunication service infrastructure has little of it. Such limited flexibility hinders progress towards new generation user services. It is argued throughout this work that a service management platform that is General, Ubiquitous, Automated and functioning as an open market can provide the necessary flexibility for a new generation service infrastructure. A *general* service management platform can manage a wide range of services using a core set of common mechanisms. Extensions are adopted as appropriate to the context of the service being delivered. A range of services is discussed in section 2.1. A *ubiquitous* platform can manage service availability over technologically and administratively heterogeneous delivery infrastructures, and finally, an *automated* approach is desired to manage the complexity introduced by flexibility.

Based on the current approach to service delivery, management of new generation services may be addressed by the following functional areas. These are grouped into three broad categories as shown in Figure 1.

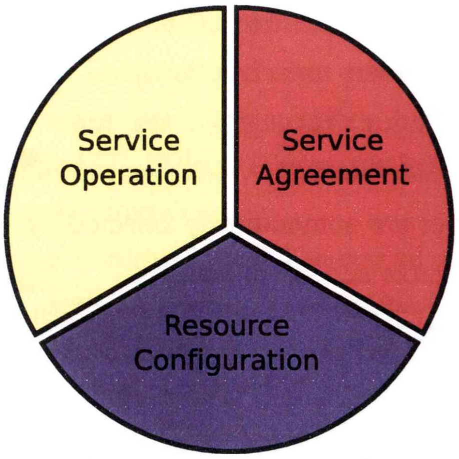


Figure 1: Service Management Components

- Formal Service Description – To facilitate automated service selection and automated resource configuration, a description of services in a machine-understandable formal language is required. The method used to describe services should be general enough to describe any current service and potential future services.
- Formal Service Level Agreement Description – If some level of commitment is required for a service delivery, the service may be described as part of a contract or *Service Level Agreement* (SLA). To maintain a manageable level of user complexity in a dynamic environment, automated SLA management is preferred. To facilitate automation, Service Level Agreements should be unambiguously described in a format that can be reasoned with by both computerised agents and humans.

  The formal description of Service Level Agreements is addressed within this dissertation. Chapter 2 outlines the current approaches, chapter 3 lists the requirements of an appropriate solution, and chapter 4 describes the SLA ontology developed by QDINE in contribution to this research area.
- SLA Distribution Framework – To form an agreement on the services and service levels to be provided, participants in a telecommunication-based service environment require a mechanism by which they can communicate. To maximise flexibility, such a mechanism should support a full range of SLAs, be independent from the service implementation and support a full range of business models for SLA distribution. The entire life-cycle of an SLA should be supported including service discovery, SLA creation, SLA modification and SLA cancellation or completion.

  To arrive at an acceptable and binding Service Level Agreement, involved parties must endorse the final proposal. Negotiation theory provides guidance in reaching such an agreement. Research into the design of negotiation mechanisms and protocols explores these issues[6][7]. A flexible SLA management platform should support any negotiation mechanism appropriate for a desired service delivery business model.

This dissertation presents an SLA distribution framework as a novel contribution in SLA management. The requirements identified for the framework are presented in chapter 3, and the QDINE negotiation framework is described in chapter 5.

- Human interface – Human participants in a flexible service management platform should have the ability to specify their desired services and associated performance levels. The platform should effectively translate user desires into the formal representation used for describing services and service agreements. This function has a large impact on system usability, but need not be common amongst participants, so is not addressed further.
- Resource Allocation – In a competitive environment, service resources should be allocated to consumers in an equitable manner. Techniques based on economic mechanisms include pricing through Markets[8], Auctions[9][10], take-it-or-leave-it deals, as well as other novel mechanisms[11]. Alternatives to economic approaches are also used[12][13]. The way in which a service provider allocates resources to consumers is specific to the service in consideration. The resource allocation method may influence the mechanism used for SLA distribution and the cost of a service. A flexible service management platform should support any resource allocation mechanism appropriate to the managed services.
- Pricing – Commonly used as a tool for resource allocation, service pricing is concerned with establishing a the *magnitude* of a cost for using a service. Service pricing works in concert with a charging model that determines *how* the costs are applied. Service pricing is an internal function performed by providers in a service delivery environment, the outcome of which may be included in an SLA for a service.

In summary, the inputs for the service agreement functional area are consumer desires, provider business models, and information about the available services. The outputs are a set of appropriate service level agreements.

Resource Configuration

Resource configuration is concerned with ensuring the resources required to deliver a service are configured so as to enable the service, and meet any commitments expressed on an associated SLA. For example, a video on demand service may involve transcoding a video into the desired format. A network connectivity service may involve establishing a virtual circuit of the desired quality between two points in a network. The resource configuration actions required to enable a service vary with the type of service to be provided. For new generation services, resource configuration should be automated to provide a real-time response. Using SLA-based service management, service providers must translate the abstract description of a service provided in an SLA to a concrete configuration for application to the service resources. The CADENUS project[14] adopts such an approach for network connectivity services, translating user SLAs into policies used for network configuration.

Automated resource configuration is a large and fertile area of research, and as mentioned above, is specific to the service in question. Some approaches used for configuration of network connectivity are discussed in section 2.3.2. For a generalised service management platform, resource configuration is an implementation concern and so is not address further. In summary, the inputs to the resource configuration function are a set of service level agreements, the outputs are a set of correctly configured resources that can provide the described services according to their SLAs.

Service Operation

The service operation group of functions manage the run-time concerns of an active service.

- Measuring – To make acceptable decisions on when to, or not to commit to a particular service level agreement, and for use in service accounting, a provider must be well informed about its environment. A provider may have a static representation of what resources have been allocated and what should be available, and a system in place to extract real data from the environment as to the actual state of the managed resources.
- Monitoring – Accurate live system measurements are a critical input to a successful service management platform. This is to ensure that committed levels of service are being delivered and that SLA violations are detected. A participating agent will attempt to minimise these violations and ultimately maintain the end user's Quality of Experience. This type of monitoring may be done by the consumer, supplier, or a nominated third party. Many approaches to automated performance measuring have been explored, as well as monitoring the compliance with an SLA[15][16].
- Charging/Billing – As services are used, the appropriate customer is charged according to the charging mechanism and service pricing. Additionally, invoices for these charges must be distributed. A provider may require that a billing path is established before a service is provided. Within current generation networks, research has suggested that the cost of providing and maintaining a mobile network billing system may be anything up to 50% of the total infrastructure investment and annual turnover[17]. Billing complexity will likely increase in a ubiquitous service delivery environment. Seamless, automated billing is therefore a large concern for

current and new generation network operators. The way in which a billing path is established need not depend on the service which is being billed; that is, a common technique can be used to establish a billing path for any service. ebXML[18] allows businesses to discover and share common business processes, and may be useful in this area.

- Payment – Once charges are distributed, a technique is required to facilitate payment. Traditional payment approaches such as credit cards or electronic funds transfer may be used, as well as micro payments or alternative value systems [19].

Although important in a service management platform, the above functions are not addressed in detail by this dissertation. This is to limit the scope of work to a manageable PhD research project.

1.4 Thesis, Contributions and Significance

My thesis is:

The use of a knowledge-based language to describe SLAs, coupled with an open service market employing autonomous intelligent agents, can provide the flexibility required for ubiquitous, generalised service delivery in a new generation telecommunication environment.

This work was conducted within the normal constraints of a PhD research project and therefore does not attempt to address all functions of a service management platform. Two key contributions are made by this work:

1. A set of formal ontologies that are used to semantically describe secure service level agreements for any application domain.
2. A multi-agent system providing an open market where services can be discovered, participants identified, and negotiation performed using context specific mechanisms.

An *open market* is adopted as it encourages the interoperability required for trade, supports autonomy and flexibility in business models, drives progress through competition, and provides consumer choice along with an efficient mechanism for resource allocation. Further contributions of this work are described in section 7.1.

The work described in this dissertation has significance to the e-business community, telecommunications industry and associated research community, as well as other service industries that have a distribution model utilising a telecommunications network. Adoption of this work within an operational service environment allows service providers and consumers to establish and manage binding Service Level

Agreements for provided services.

1.5 Outline of the Dissertation

The remainder of the dissertation is structured as follows:

Chapter 2 reviews the main components of the problem domain: An analysis of telecommunication-based services, current approaches to describe service agreements, resource allocation, service pricing, and service ubiquity. Chapter 2 finishes with a critique of the current art.

Chapter 3 discusses the requirements for addressing the two functional service management areas covered by this work, and a solution proposal that introduces chapters 4 and 5.

Chapter 4 proposes a set of ontologies that may be used to describe service level agreements in any service domain. A concrete formalism is provided and the key classes in the ontologies are discussed.

Chapter 5 then describes a negotiation framework for the distribution of service level agreements to consumers in an open market. The framework is a multi-agent system addressing the communication model, social behaviour of agents, and the market processes.

Chapter 6 describes the techniques used for validating the proposed approach. Satisfaction of the solution requirements is discussed, an analysis of the choices made in deriving the proposed solution is given, and the proposal is compared with the current art. Chapter 6 concludes with a description of an experimental system and use case.

Chapter 7 concludes the dissertation with an outline of the key components developed, the opportunities for future work and offers some concluding remarks.

1.6 Typographic Conventions

In order to symbols with special meaning in the text, a number of different fonts are used:

- An *italic* typeface is used for concepts which are immediately followed by a definition, for key quotes, or when special emphasis is to be placed on a word.
- A `fixed-width courier` typeface is used for source code and whenever objects from source code are referenced in-line. OWL class and property names use `camelHumpMultipleWordNotation` and additionally, class names have an

`InitialCapital`.

1.7 Assumed Knowledge

The reader is assumed to have a general knowledge of telecommunications, computer-based agents and a basic knowledge of XML. When new concepts are introduced throughout the dissertation, references to key literature are provided, where the reader can learn more about the addressed concepts.

INTENTIONALLY BLANK

Chapter 2
Background to Services

The primary purpose of a telecommunication network is to deliver one or more services to a consumer. Additionally, the telecommunications network can itself be considered as a service. To ground the work described here, a fundamental description of services is provided.

The purpose of this chapter is to establish a foundation of work on which the QDINE solution is built. Telecommunication-based services are first analysed from three view-points. The remainder of the chapter explores the domains of knowledge involved with service negotiation and the creation of service level agreements for services delivered over telecommunication networks.

2.1 What are the Services?

Services discussed in this dissertation are the group of network-dependant services, they are a subclass of all non-tangible things that one entity can provide to another. In this dissertation, *service* is defined as:

A functionality provided by one entity to another, either directly or indirectly, which either provides or utilises a telecommunication network.

The analysis of services for telecommunication networks is approached from three orthogonal standpoints:

- Services are analysed temporally, considering the history of services, the services available now such as Voice over IP, and possible future services.
- The hierarchy of services is examined, exploring service inter-dependence and the place of each service type in the layered network paradigm.
- The operation of services is considered, exploring the difference between connection-oriented and connectionless services, service sessions, and a comparison of client-server and peer-to-peer delivery.

2.1.1 The Temporal View

The roots of telecommunication lie in the telegraphic transfer, the first electric telegraph was demonstrated in 1837 by Wheatstone & Cooke[20]. For around forty years, the telegraph was used to transmit information, encoded as pulses of variable duration across an electric transmission line. The telegraph gave way to the beginnings of modern telecommunication networks in 1876 when Alexander Graham Bell first sent voice over an electric wire. This was important, as it did not purely involve a binary on and off signal state on the wire, but accommodated the essential spectrum of human voice. A few years after this discovery, the telephone, which has changed little to date gained popularity.

Even in the embryonic stage of telecommunications, a range of services had begun to emerge. At the most basic level, there was a connectivity service of the actual physical lines, then a range of telegraphic services such as Morse Code, Baudot Code, the Bain Facsimile, teleprinters and numerous other proprietary information transmission methods such as the Western Union Varioplex[21]. Telephony, however, was the prime growth service.

The invention of the telephone and its analogue transmission method led to a host of other services, leveraging the bandwidth analogue transmission provided. Arthur Korn arrived first with the photoelectric facsimile in 1902 that was able to send greyscale photographs across a wire. Later followed the Belinograph, AT&T Wirephoto, the Hellschreiber and the RCA radiotelephoto among others.

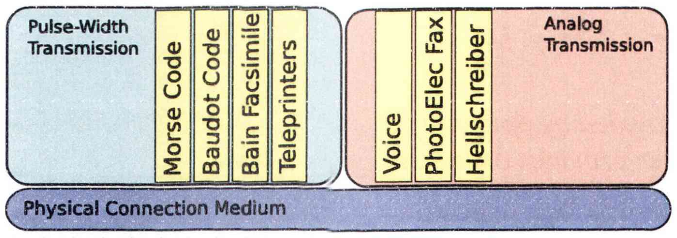


Figure 2: Two basic service layers

Telecommunication services are delivered over a physical medium. Electrically conducting wire was the physical medium used by Wheatstone and Cooke in 1837 and is still widely used today. In 1866, Mahlon Loomis, a dentist from the USA, successfully demonstrated the concept of wireless telegraphy, and correspondingly a new transmission medium; the electro-magnetic (radio) wave. The invention of the radio telegraph followed later in 1895, adopting this transmission medium.

In the late 1940s, the transistor device was invented, paving the way for modern computing. It was not long before scientists began to combine telecommunications and computers, and in the 1950s, North American defence scientists as part of the Semi-Automatic Ground Environment (SAGE) project [22] developed the first modem for use in transmitting digital information over long-distance analogue telephone lines. This first interface between digital and analogue transmission modes heralded a new leap forward in telecommunications.

Prior to the modem, there had been two layers of service. The connectivity layer, and a layer of single function services operating on top of the physical connectivity. The ability to connect computers over telephone lines, coupled with the multi-functional nature of computers, granted services a further layer of separation from the underlying physical network. It was now possible to perform a range of functions over the same underlying digital modem link (Figure 3). This "Digital Access Layer" was analogous to the physical connectivity layer in that it provided a distinct platform on which further services could be built.

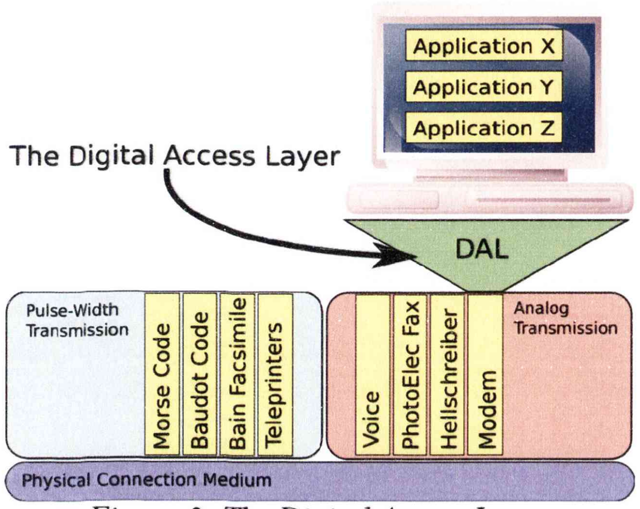


Figure 3: The Digital Access Layer

An attempt to build a survivable communications network[23] was begun in 1962 within project RAND[24]. *Survivability* is a measure of network node inter-operability after a physical attack on the network. This was proposed through the use of packet switched networking. *Packet Switching* is a telecommunications data transfer technique whereby messages are divided into units called packets. Packets can be routed from their source to any destination within a set of inter-connected communication links. Each link is terminated by a packet switching node. Prior to packet switching, communication was performed over end-to-end, circuit switched paths. Under *circuit switching*, the destination for data injected onto a communication link is determined by network configuration, not by information contained within the sent data. With circuit switching, an end-to-end connection is established prior to data transfer. To change the destination, the connection has to be physically re-established to the new destination.

By 1969, packet switching technology had sufficiently matured from concept into a working implementation when the foundations of the current Internet were laid with the beginning of ARPANET[25]. In 1970, the first "standard" packet switched protocol was released as the Network Control Program (NCP)[26].

The dawning of packet switched networks was a fundamental step from the viewpoint of services. Packet switching provided another layer of abstraction from the physical network, allowing physical links to support different types of applications, to different destinations *concurrently*. However, each end host on the network could only use its network connection for one application at a time.

From 1970 to 1973, numerous applications were developed to use the new packet based networking. Most notably, Telnet and FTP in 1971, these were followed by the first email program written by Ray Tomlinson in 1972.

1974 saw the publication of the founding paper for the TCP/IP communication protocol suite [27]. Here, the concept of sockets was introduced whereby many processes running on a specific network host could communicate simultaneously with many other processes on remote hosts. A TCP standard was proposed later that year as RFC 675 [28]. This concept of sockets enabled yet another layer of separation of the physical network from the end user applications.

Prior to 1974, digital networks were either limited to a local area, or were carried as an overlay on the analogue telephone network using modems. As more services employing modems began to operate, scientists developed encoding such as manchester[29] to send digital signals for greater distances directly over an electrically conductive medium. The X.25 protocol suite[30] was the first widely accepted technology for digital Wide Area Networking (WAN). X.25 is a packet switched networking protocol suite and was deployed in 1974 when Sercnet[31] was established as the first X.25 based digital network. Telecommunications Companies realised the benefit in packet based networking and began digitising voice to transmit it over X.25 networks. X.25 has very low data transmission and is not commonly used today. ISDN[32] and Frame-Relay [33] are descendants of X.25 and are still in common use.

The Integrated Services Digital Network (ISDN) is a circuit-switched telecommunication system allowing the transmission of digital voice and data over a single twisted-pair telephone line. An ISDN connection consists of one control channel (Up to 64kbps) and up to 30 data channels, each with 64kbps bandwidth providing a maximum bandwidth of 2.048Mbps. The goal of ISDN was to bring digital networks into common use.

Frame Relay is a packet-switch telecommunication system with the aim of offering users a permanent virtual circuit between two end points. The end-user can treat the frame relay connection as it is a permanent circuit-switched path, however data sent over a frame-relay network is divided into units called frames and the physical path taken by frames may vary over the life of the connection. Frame-relay is targeted to businesses who may require a point to point connection for a long duration.

By the late 1970s, the basic structure of network related service types(Figure 4) had been forged and has changed little to date. Advances in technology and services mainly involved new applications, different protocols, faster speeds, higher reliability or greater mobility. Today, when telecommunication related services are discussed, the following concepts may be mentioned.

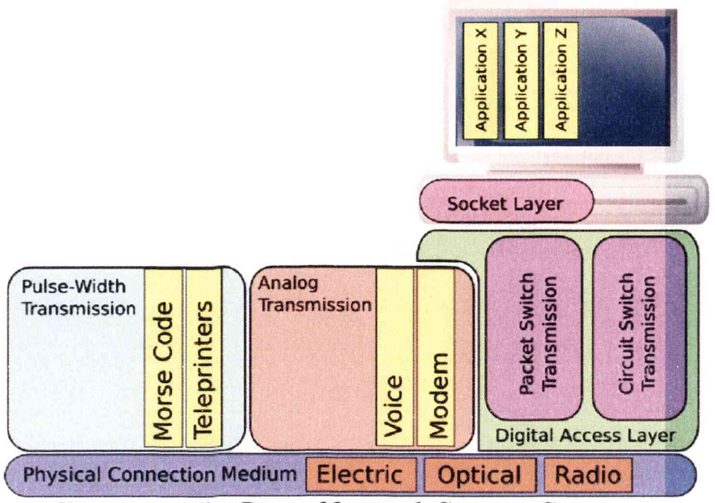


Figure 4: The Basic Network Service Structure

Land Line Phone Call, Cellular Phone Call, SMS, e-mail, the Internet, ADSL, ISDN, VPN, frame relay, MPLS path, Skype, VoIP, MSN Messenger, GPRS

Any one of the above concepts may be described as a service. This dissertation views all the above as network related services, however it is by no means a definitive list. We now look to the future direction of services.

New telecommunication functionality available to services may be driven by user demand and are hence *pulled* into existence. Alternatively, a new capability may be identified by a provider and then *pushed* within a service into the market. Most successful telecommunication services are pulled by demand (for example, a survivable network or the ability to multi-task). This demand filters from the users, through the applications they use, down into the application support services. Numerous attempts have been made to push functionality onto the network for the network users, hoping applications and the inquisitive user will adopt the advances in technology. This approach has had limited success. An example of such a failed approach was the introduction of WAP functionality on Mobile Phones [34].

A successfully delivered telecommunication function that was pulled by demand is that of connection mobility. People want to stay connected: the more available a connection, the better. The possibility of untethered connections allows connectivity where previously not possible. A step beyond, is enabling user mobility for an untethered connection. The demand for such mobility is evident through the success of cellular phones. 13 of the 30 OECD countries have a mobile subscription rate greater than 95 percent[35]. The *mobile subscription rate* is the number of mobile phone subscriptions active in a country per head of population. In today's massively connected world, some form of telecommunication network can be accessed from almost any point on the globe, albeit with varying levels of reliability, performance and cost. Very few of these networks work together to provide uninterrupted connectivity for a mobile user, resulting in intermittent and patchy connections. The aim of ubiquitous connectivity is the integration of telecommunication networks into a seamless fabric of connectivity.

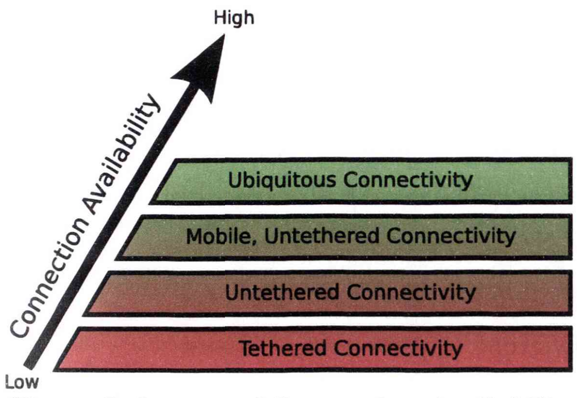


Figure 5: Increased Connection Availability

In the current "Internet Age", user demands are evolving rapidly. People no longer accept that something can't be done. If someone can't provide a service, a competitor will be found who can. In this environment, applications and their demands on the network also rapidly evolve. Two areas in which this can currently be seen is computer gaming and entertainment. Virtual worlds[36] is a relatively new technology and has the potential to place strong demands on a network in terms of dynamicity, reliability and network load.

Following history, a future for telecommunication-based services can be seen where applications and their demands from a network continue a rapid evolution, connectivity becomes seamless and ubiquitous, and services are dynamically provisioned and flexible to support rapidly evolving application requirements. Flexibility and agility of services is key to handling unexpected evolution of applications.

2.1.2 The Hierarchical View

Telecommunication-based applications may be divided into User Applications and Application Services. *User Applications* are software processes running on an end user device, interacting directly with the user. *Application Services* are software processes running on a network host and are designed to be controlled via the network. Functionalities required to support one or more applications are regarded as *Application Support Services*. For network services, these support services may be divided into providing the physical connection medium, and operation of the protocols that utilise the physical medium. The hierarchy of applications and services is further described by the three layer diagram of Figure 6, and may be illustrated by the following example of

Video on Demand. Here, the User Application may be Windows Media Player, the Application Service is the software process running on the remote video server. The physical connection media is the combination of physical hardware which connects the user to the remote software process, such as an ADSL modem, Internet routers and optic fibres. Finally, a variety of support services are provided over that hardware to carry the video. These may include the ADSL edge connectivity service to the user and any number of Internet backbone protocols which carry the traffic.

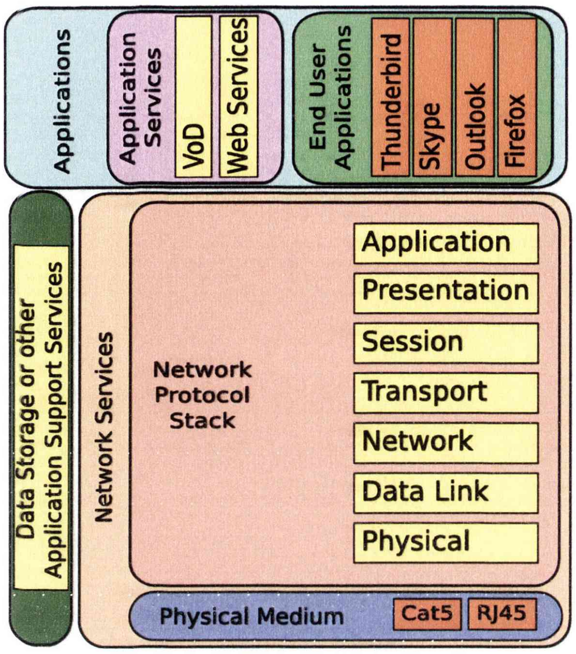


Figure 6: The Service Stack

This dissertation ignores business models that provide physical connection media or end user applications, as they require some form of physical interaction to provision and are therefore out of scope. Three examples of such businesses are: an Internet kiosk, where the user is effectively renting the hardware and application as well as the telecommunication services required to run it; an equipment vendor renting equipment to a network operator; or the rental of dark fibres to a network operator.

The services considered in this work are shown in yellow in Figure 6. These are Application Support Services and Application Services.

The Open Systems Interconnection (OSI) model [37][38] is a standardised, abstract model of telecommunications protocols. It is a layered model, designed to separate the functionality provided by a network into seven functional service layers. The model separates issues such as connection and link specifics, network addressing, transport reliability and data presentation into discrete layers, allowing interoperability with protocols from neighbouring layers. Interactions between layers in the OSI model occur through a set of standard "primitives". These primitives are functions that allow vertical communication between layers in the OSI stack. For example, the `T-CONNECT.request` primitive allows the (T)ransport layer to request a connection from the network layer.

As network protocols and functions have matured, the discrete layers of the network protocol stack have begun to blur, with protocols such as GMPLS operating at multiple levels. Table 1, adapted from [39] highlights some example protocols within each layer of the OSI model.

Layer		Misc. examples	TCP/IP suite	SS7	OSI suite	IPX suite	UMTS
#	Name						
7	Application	HL7, SIP	HTTP, FTP, Telnet,	TUP, TCAP	X.500, DAP		
6	Presentation	ASCII, EBCDIC, MPEG	XDR, SSL, TLS		ISO 8823, X. 226		
5	Session	Named Pipes, NetBIOS, SAP, SDP	TCP Session establishment		ISO 8327, X. 225	NWLink	
4	Transport	NetBEUI	TCP, UDP, RTP, SCTP		TP4, OSPF	SPX, RIP	
3	Network	NetBEUI, Q.931	IP, ICMP, IPsec, ARP, RIP, BGP	MTP-3, SCCP	X.25 (PLP)	IPX	RRC
2	Data Link	Ethernet, 802.11, FDDI		MTP-2	X.25 (LAPB)	IEEE 802.3	MAC
1	Physical	RS-232, 10BASE-T, SONET, DSL,		MTP-1	X.25 (X.21bis)		PHY

Table 1: Example protocols within the OSI Seven Layer Model

An important factor in considering the delivery of telecommunication services is determining at which layer in the protocol stack the administrative control is relinquished by the end user to the provider of lower layer services. For example, in a cellular telephone application, the service provider controls the entire protocol stack, up to and including the application layer. Details such as the choice of voice codec and session signalling method are all dictated by the service provider. In comparison, consider an email application used by the customer of a domestic Internet service. Here, the administrative control is relinquished further down the protocol stack. To the Network layer, the user is free to adopt any Transport, Session, Presentation or Application layer protocols it desires, using any application that relies on these protocols.

2.1.3 The Operational View

Service may be categorised in accordance with the way in which they are delivered.

Continuous services are one time activated with no sub level of activation. An example of a continuous service is a subscription to the receipt of stock prices via SMS to a mobile phone. This service has no meaningful sub-schedule to a service provider or end user. Alternatively, *session based services* have one or more sub-schedule or activation periods for the service. Examples of these services are phone calls, video conferences or a Video on Demand service where each movie could be seen as a session.

Orthogonal to the Continuous/Session-based dichotomy is the separation of services into either Peer-to-Peer or Client-Server delivery methods. Within a *Peer-to-Peer* service, end-users are the active participants in the service, and each generally performs a similar function, having mutual interdependence. Service providers for peer-to-peer services generally assume a passive role, providing any technologies required to support the peer-to-peer application. Two examples of peer-to-peer services are Kazaa[1] and BitTorrent[2]. Within a *Client-Server* delivery model, the server performs a dominant role, and the clients are dependant on the server to perform some function. The server is not directly dependant on any particular client. Within the client-server model, the service providers typically operate the server, and provide a direct functionality to the service user acting as the client. An example of the client-server model is a Video on Demand service.

Specific to telecommunication connectivity services, a service classification of importance is the distinction between connection-oriented and connectionless services. A *connection-oriented* network service is one in which traffic is identified as belonging to a particular flow by the providing network. A logical connection is formed from source to destination before data transfer occurs. A *connectionless* network service requires no prior connection between source and destination. Connection-based services are typically only provided when the traffic cost to set up the connection is justified by a favourable comparison with the benefits provided and the total traffic volume for the flow. Circuit switched network services are connection-based, while packet-switched network services are generally not.

2.2 What is Quality?

New generation service management requires the knowledge of *what* service is to be delivered, as well as the *level of quality* at which the service is to be delivered. The Oxford English Dictionary[40] describes quality as "a degree or grade of excellence". When discussing service quality, two aspects to quality emerge. User-focussed "quality of experience", as well as "quality of service", which is directly related to the technical aspect of service delivery.

2.2.1 Quality of Experience

Quality of Experience (QoE) is "the level of satisfaction with a service from the perspective of a user based on their needs, wants and expectations"[41]. A user has an expectation of a service to satisfy his or her specific wants and needs. The quality of experience whilst using a service may not rest entirely on the performance of the

1 http://www.kazaa.com – Accessed 12/02/2007

2 http://www.bittorrent.com – Accessed 12/02/2007

provided service: A user may have incorrectly chosen the service in regards to his or her wants or needs, or a device accessing a service may impact the quality of experience.

The work described here deals specifically with the aspects of QoE that can be addressed through describing the expected performance from a technological viewpoint, that is, through quality of service metrics. The maintenance of expected service quality has a direct impact on a users quality of experience.

2.2.2 Quality of Service (QoS)

A service may have one or more characteristics that influence the perceived quality of experience to an end user. These characteristics can be used to describe a relative service quality amongst services of the same type.

If an end user can specify his or her expectation of a service using these service quality characteristics or "QoS Parameters", the service provider has a well defined target for user satisfaction.

Different service types have different characteristics, specific to the functionality they attempt to provide. There can therefore be no definitive list of QoS parameters for all possible services. Each type of service will have parameters appropriate to its purpose, and each new service may introduce new parameters. For example, a Video on Demand service may have resolution as a QoS parameter, whilst an IP connectivity service may have throughput as a parameter. A service description should specify the QoS parameters applicable to that service.

Services discussed in this dissertation all rely on, or provide some form of telecommunication network. QoS parameters applicable to telecommunication networks are therefore discussed below. Four important service characteristics used within telecommunications and applicable to most connectivity services are described.

2.2.2.1 Connectivity Service Quality Parameters

The following four widely accepted connectivity service performance parameters are generic and can be applied to any packet-based telecommunications network:

- Packet Loss
- Packet Delay
- Throughput
- Jitter

An end-to-end telecommunications path usually traverses multiple networks, and a unit of traffic will likely transit multiple networks consecutively to reach its destination. The aggregation properties of the four basic performance parameters are discussed for data traversing a sequence of multiple networks. The equations described are based on queuing theory and an assumption of independence between networks.

2.2.2.1.1 Packet Loss

Packet loss in a network is the disposal of a data packet during transit before arriving at its destination. Loss as a service parameter is usually expressed as a percentage. Aggregate loss from traversing multiple networks is a product of the individual losses experienced in each network, and is defined below:

$$L_{Total} = 1 - \left(\prod_{i=1}^{n} \left(1 - L_i \right) \right)$$

Formula 1. Total percentage loss through a Series of Networks

L_{Total} is the total loss across the end-to-end path through n networks and L_i is the percentage loss through the i^{th} network in the series. Such that $L_i \in [0,1]$

Within current wired telecommunication networks, loss is typically caused by the over utilisation of a network link, resulting in the discarding of packets. Within wireless telecommunications, loss is often caused from interference on the radio interface.

2.2.2.1.2 Packet Delay

Packet delay (or *latency*) is the end-to-end trip time for a packet travelling from source to destination. Delay through a series of networks is cumulative. The aggregation formula for series network delay is as follows:

$$D_{Total} = \sum_{i=1}^{n} D_i$$

Formula 2. Cumulative Delay through a Series of Networks

D_{Total} is the total delay across the end-to-end path through n networks, and D_i is the delay experienced by a packet through the i^{th} network in the series.

Packet delay in a network is can be attributed largely to four factors:

- Propagation delay
- Transmission delay
- Queuing delay
- Processing delay

Propagation delay is caused by physical distance the signal must travel from source to destination and is constant for a given route and transmission medium. Transmission (or Serialisation) delay is a function of the transmission rate of the network, for example, an 8kb packet will take 8 seconds to serialise over a 1kbps link, but the same packet will take only 0.0125 seconds on a 1Mbps link. Telecommunication equipment usually has a buffered outgoing interface. Queuing delay is a result of the time a packet spends in the outgoing buffer waiting to be sent. It is the most controllable of the delay factors and can be adjusted by varying the outgoing queue length and scheduling algorithms. Queuing and Transmission delay comprise the majority of delay experienced by a unit of data. Processing delay is determined by the speed at which the networking equipment can process incoming data and place it into an outgoing queue.

2.2.2.1.3 Throughput

Throughput of a network path is the volume of data transferred between source and destination within a set time period, usually one second. Often measured by the receiving rate at the destination. Aggregate throughput over a series of networks is governed by the weakest network in the series. That is, throughput is non-increasing with the series aggregate equal to the minimum throughput of any network in the series.

$$T_{Total} = min\{T_i\}_{i=1}^{n}$$

Formula 3. Throughput experienced through a Series of Networks

T_{Total} is the throughput experienced across the end-to-end path through n networks. T_i is the throughput over the i^{th} network.

2.2.2.1.4 Jitter

Network jitter or Inter Packet Delay Variation (IPDV) is the deviation in arrival times of packets from their scheduled arrival times. The maximum jitter over an end to end link through a series of networks is the sum of the IPDVs for each network in the series. The actual jitter for the end to end link, however, will typically be less than this maximum figure as the variation may be in either a negative or positive direction. Variation in the negative direction for a packet through one network may be compensated for by variation in the positive direction through a following or previous network. Aggregate jitter is expressed as follows:

$$J_{Total}=\sum_{i=1}^{n} J_i$$

Formula 4. Total jitter through a Series of Networks

J_{Total} is the jitter experienced across the end to end link through n networks. J_i is the jitter through the ith network in the series.

Two major causes for network jitter are from packets belonging to the same flow travelling via different routes to a destination or an inconsistent queuing time for packets.

2.3 Network Quality of Service Methods

Services discussed in this dissertation all depend on, or provide a transport network. If no quality assurance is provided by the network, the quality of the entire service can not be guaranteed.

Two considerations in enabling network QoS are: the techniques employed by network components to deliver quality controlled traffic, and the methods used to configure these network components for the quality delivery.

2.3.1 Network QoS Techniques

All network QoS techniques share one broad goal: To provide some certainty on the level of performance for traffic passing through a network. These techniques may employ *Traffic Differentiation,* which enables discriminatory treatment of traffic, *Admission Control,* which limits the number of users and consequently the load on a resource, and *Constraint Based Routing* which allocates traffic to routes depending on the current performance of the route.

2.3.1.1 Bandwidth Over Provisioning

Over Provisioning is the process by which the resources made available to a network are greater than the average load. Usually many times the required resources are made available. Traffic differentiation is not supported.

Although not widely considered a QoS technique, bandwidth over provisioning has the aim of providing a probabilistic Quality of Service for traffic. Over provisioning becomes more appropriate as load aggregation increases and the proportion of an average single unit load becomes smaller in relation to the maximum capacity of the resource. In this scenario, the statistical average load of all flows is stable and the impact from the load variation of one flow on the QoS of all flows is low.

Given the small set of applications with modest demands that have typically dominated the Internet, an Over-Provisioning approach has been somewhat appropriate. However, research suggests that over provisioning is not appropriate for future telecommunications[42] where many more services with firm, complex and conflicting requirements will be aggregated into a common network. As the proportion of real-time services increases on a network, the level of over-provisioning must increase for an acceptable service quality to me maintained. Bandwidth over-provisioning is therefore not considered a viable long-term solution as Internet traffic increases rapidly every few months.

2.3.1.2 The Differentiated Services Framework

The *Differentiated Services Framework* (DiffServ)[44] is a traffic classification framework designed by the IETF to work with IP networks. On entry to a DiffServ enabled network, a packet is assigned to a traffic class by being marked with a particular label or Differentiated Services Code Point (DSCP) within the DS Field in the IP packet header(Figure 7). The code assigned to a packet is determined by a traffic classification algorithm and generally uses operational factors such as protocol identifier, port number or source and/or destination address. At each router inside a DiffServ network, each packet is treated according to the Per Hop Behaviour (PHB) associated with the packet's assigned DSCP. A Per Hop Behaviour defines a relative traffic treatment in terms of resource priority relative to other PHBs (e.g., buffer, bandwidth) and in terms of relative observable traffic characteristics (e.g., delay, loss). An important characteristic of DiffServ is its scalability. As such, it is commonly used in the network core.

```
Bit#    0   1   2   3   4   5   6   7
       +---+---+---+---+---+---+---+---+
       |         DSCP          |  CU   |
       +---+---+---+---+---+---+---+---+

DSCP:  differentiated services codepoint
CU:    currently unused
```

Figure 7: The IP Header DS field [43]

The DiffServ mechanism offers no strict guarantees on the characteristics of traffic. It enables probabilistic guarantees through relative traffic treatment for a single link within a network. The status of links, such as current load or performance is not distributed throughout the network and each link operates independently when treating traffic. For stronger, deterministic guarantees, IntServ may be employed.

2.3.1.3 The Integrated Services Architecture

The *Integrated Services Architecture* (IntServ)[45] was designed to support the integration of demanding multimedia services onto a common network. As such, IntServ provides strong deterministic guarantees to network traffic. Traffic is assigned to specific flows and resources are reserved at each router for every active flow. IntServ capable devices in an IntServ network maintain the state of each flow carried over their interfaces. Strict admission control practices are maintained for an IntServ Network ensuring existing reservations are not disturbed by newly admitted traffic.

Every IntServ flow has a traffic specification stating the characteristics of the traffic. This is the profile of the data flow in such terms as rate and burstiness. Additionally, each traffic path is assigned to one of three three classes of service; guaranteed, controlled load or best effort. These service classes describe the strength of the guarantee required by the traffic flow and allow differentiated treatment of traffic classes.

The IntServ architecture employs the Resource reSerVation Protocol (RSVP)(§2.3.2.4), using PATH and RESV messages for signalling resource availability, requests and reservations. Due to each router's awareness of supported flows, IntServ has inherent scalability issues and is therefore inappropriate for the Internet backbone. For example, consider an Internet core network link with a bandwidth of 50 gigabits per second (Gbps). The adoption of IntServ with 512kbps average flow size, results in approximately 100,000 managed flows!! That is an enormous management overhead for routers.

2.3.1.4 802.1p/d

802.1p (later integrated into 802.1d[46]) is a protocol extension applicable to all IEEE 802 MAC protocols (Ethernet, Token Ring, DQDB) and ANSI-based FDDI. It allows the assignment of link layer frames to traffic classes for discriminatory treatment similar to DiffServ. Eight different classes of service are available, expressed through three extra bits on the Ethernet Frame. The manner in which traffic is treated when assigned to any particular class is undefined and left to implementation. The IEEE however, has made some broad recommendations in the 802.1d specification.

2.3.1.5 Asynchronous Transfer Mode

Asynchronous Transfer Mode (ATM) is a connection oriented packet switching and multiplexing technique for Broadband-ISDN that directs traffic using a connection identifier field contained in the packet header. An end-to-end virtual circuit of specified bandwidth is created for data transmitted over the ATM network. ATM is not specifically a QoS technique, but it is a network transport protocol that has inbuilt QoS features. This is achieved through admission control, traffic differentiation and constraint based routing.

Unlike other packet transport technologies such as IP or Ethernet, ATM uses short, fixed length packets called cells. Each cell is 53 bytes long: 48 bytes for the payload and 5 bytes for the preceding header and connection identifier. The connection identifier is usually composed of a Virtual Path Identifier of up to 12 bits and Virtual Circuit(VC) Identifier of 16 bits. Every ATM transmission link may include 4096 virtual paths with each path containing up to 65536 virtual circuits.

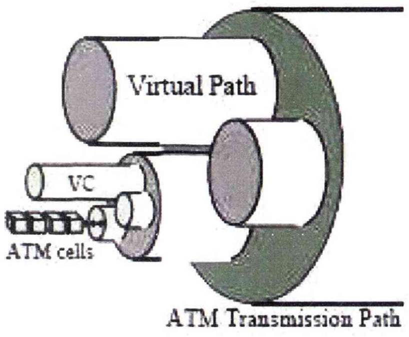


Figure 8: ATM Hierarchy

To establish an ATM virtual circuit, a signalling process is first completed to determine the Quality of Service to be given to that particular virtual circuit. Five QoS classes or traffic types are available to ATM circuits. In decreasing levels of guarantee, these are: Constant Bit Rate (CBR), real-time Variable Bit Rate (rt-VBR), non real-time Variable Bit Rate (nrt-VBR) and Available Bit Rate (ABR) and Unspecified Bit Rate (UBR). Each of these traffic types has performance attributes such as Peak Cell Rate (PCR) and Minimum Cell Rate (MCR). The values of which can be individually defined for each virtual circuit carried by the network.

2.3.1.6 (Generalised) Multi-Protocol Label Switching ((G)MPLS)

MPLS[47] is a label switching protocol similar in operation to ATM. It is designed to integrate ATM and Frame Relay with IP. In MPLS, the path taken by labelled IP packets is based on the label stored in a local forwarding database rather than on the destination IP address, eliminating the need for prefix matching, performed by IP routers. Faster lookups, combined with fewer labels to examine than IP addresses in a router, makes label switching faster than regular IP forwarding. GMPLS uses the same mechanism as MPLS, extending it to add generalised support for other paths that ATM such as optical wavelengths or time slots in TDM.

To establish an MPLS path, an initial signalling process is first completed to determine the label to assign to IP Packets entering the MPLS network. The Label Distribution

Protocol (LDP)[48] or RSVP(§2.3.2.4) can be used for MPLS signalling.

ATM allows the use of explicit routing, building quasi-connections for a particular flow. The possibility of explicit routing allows the assignment of traffic to specific links and the ability for traffic engineering (TE) and QoS extensions to MPLS, referred to as MPLS-TE. This traffic engineering is delivered through Constraint based Routing and/or coupling DiffServ with MPLS.

To provide support for constraint based routing with MPLS, traffic engineering extensions to RSVP have been developed: RSVP-TE[49], along with extended versions of routing protocols such as OSPF-TE[50] and ISIS-TE [51]. DiffServ-TE[52] outlines the interaction between MPLS and DiffServ for providing DiffServ based QoS support for MPLS.

2.3.2 Network Configuration

Network Configuration is the process of adjusting the hardware and software set-up of network components to apply the appropriate QoS techniques for specific traffic. Here, we address the provisioning mechanisms which may be used to configure a network device to deliver a quality controlled service. The scope of this discussion is limited to configuration mechanisms which can be used remotely; that is, where physical access to the device is not required.

Equipment vendors each have proprietary methods for accessing and configuring their equipment. Additionally, efforts have been made for standardised configuration interfaces. Sections 2.3.2.1, 2.3.2.2 and 2.3.2.3 describe *Centralised Configuration* approaches where some management entity has knowledge of the entire network and may configure devices in consideration of that view. The final four approaches consider *Distributed Configuration,* where local decisions are made concerning the configuration of the network.

2.3.2.1 Proprietary Methods

Early implementation of remote device configuration was through a Command Line Interface (CLI). Access to a network device via a CLI is commonly achieved through a Telnet session, operating in-band over the data channel. CLIs are still widely used for device configuration today, as they are robust and operational knowledge within the network administration community is high.

Additionally, equipment vendors typically support device configuration through a proprietary set of Application Programming Interfaces (API)s. These APIs are often made available over proprietary protocols, Remote Procedure Calls or optionally over a common middleware such as Corba[53]. *Middleware* is software that connects two or more software applications so that they can exchange data. These APIs are often used with vendor specific management tools, designed to ease the task of network administration.

Many equipment vendors provide interfaces to device configuration using web technologies; often as an embedded web server running on a network device and hosting a specific mini web site as a graphical interface to the host device configuration. The Alcatel OmniPCX Communication Exchange[54] offers a Web Service API for operating the exchange, however a standardised set of web services for configuring devices from any vendor is not available. For a further discussion on Web Services, see Section 2.7.3.1.

Proprietary configuration mechanisms make the task of network administration difficult in a multi-vendor environment. For this reason, standard configuration interfaces have gained popularity.

2.3.2.2 Simple Network Management Protocol (SNMP)

The *Simple Network Management Protocol* is a standardised protocol allowing access to network device configuration information and operational statistics. SNMP is an ISO layer 5 protocol, with mappings defined for numerous transport protocols. The only mandatory support for a transport protocol is for UDP over IPv4. Device operational and configuration information are both made available via Management Information Bases (MIB). An MIB is an information database with a standardised interface, available from every SNMP capable device. It defines the properties of a set of managed objects within the device. Each managed object defines a standard interface to a function provided by the device. Properties of the managed objects provide both real-time and aggregate performance data and enable configuration of long term device information.

Retrieving data, or configuring an SNMP capable device is achieved by querying or setting one of the object properties contained within the device's MIB. Properties are either read only or read/write capable. The interfaces provided by MIBs are mostly standardised for common functionality, such as the IP Forwarding Table MIB[55], however vendor and device specific MIBs do exist for accessing extended functionality[3,4].

SNMP is typically used by network administration and monitoring tools for interacting with equipment from multiple vendors operating within a single administrative domain. It is a hierarchical management approach, using remote monitors, however there is one centralised root where decisions are initially made and changes propagated to devices via the remote monitors.

2.3.2.3 Policy-Based Network Management (PBNM)

Policy-Based Network Management is a standardised platform for applying administration policies concerning users and applications to specific network actions, including bandwidth management, security, caching, routing and VPN creation.

A policy-based networking system defines two main components: a policy decision point (PDP) and policy enforcement points (PEP). A PEP controls a network resource and configures the resource in accordance with directives given by a PDP. A PDP makes decisions based on policies entered by a Network Administrator. Policies used within PBNM consists of a condition and an associated action. Whenever a PEP encounters a condition for which it has no decision state, it requests from a PDP a decision based on the new condition. If a PDP has given a decision to a PEP and the decision becomes invalid through a policy change at the PDP or a change in network state, a PDP can retro-actively modify an existing decision given to a PEP without a preceding request from the PEP. The signalling protocol COPS (Common Open Policy Service) is used to communicate requests and decisions between policy enforcement points and policy decision points.

PBNM operates in a centralised function. The policies for multiple devices are entered into a central repository by a Network Administrator where they are accessed by PDPs in the network. PBNM has a defined relationship with the Resource Reservation Protocol (RSVP)(§2.3.2.4), where PBNM policies can be used to guide decisions for RSVP requests.

2.3.2.4 RSVP

The *Resource ReSerVation Protocol* (RSVP)[56] is a signalling protocol, originally designed for use with the IntServ framework. It is a real time, multi-hop resource reservation protocol for establishing reserved communication paths from source to

3 ftp://ftp-sj.cisco.com/pub/mibs/v2

4 http://www.juniper.net/techpubs/software/junos/junos76/swconfig76-net-mgmt/html/snmp-mibs.html

destination within a telecommunications network.

In operation, a traffic source node requests a resource reservation to one or more destinations. The request proceeds in multiple hops towards the destination. Each RSVP capable router in the path checks if it can satisfy the request before forwarding the request to the next router towards the destination. On arrival at the destination, path confirmations are sent from the destination to the source. RSVP works on a soft reservation model, whereby if the bandwidth isn't re-reserved periodically, it is released. RSVP is not concerned with routing, only with reservation over previously defined routes. RSVP is not directly concerned with QoS, but carries as part of the reservation request a "flowspec" where QoS parameters for a flow are specified. In addition to the flowspec is a filterspec which is used to identify the flow. Each router in an RSVP path inspects this flowspec which is sent to an "admission control" module in the router. That module decides if the request can be fulfilled. The "admission control" module is outside the scope of the RSVP protocol and would be handled by a traffic engineering system such as DiffServ or IntServ. In addition to the request/reply function described above, a router can additionally send capability messages to its neighbours via advertising (adspec) messages.

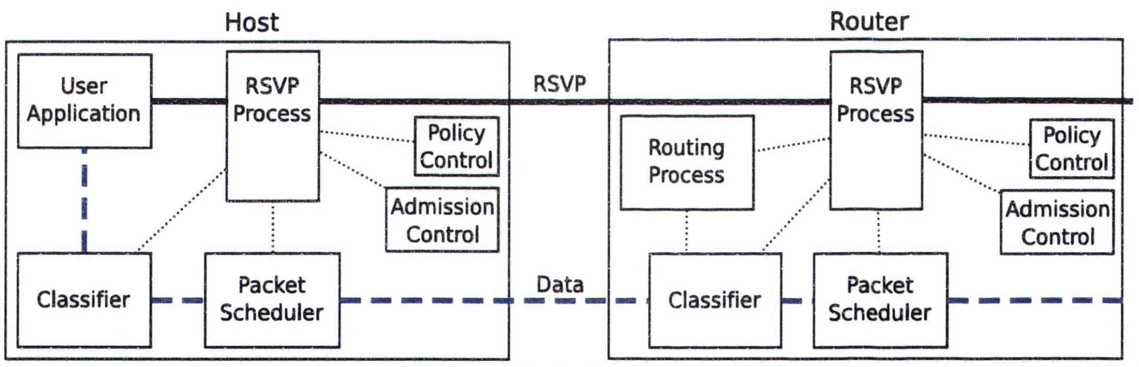


Figure 9: RSVP Components (Adapted from [56])

Traffic engineering extensions [49] have been defined for the RSVP framework to support MPLS traffic engineering requests as described in section 2.3.1.6.

2.3.2.5 QBGP

QoS-enhanced Border Gateway Protocol (QBGP) is an extension to the inter domain routing protocol BGP with the addition of link level quality of service parameters. BGP uses a flooding algorithm to distribute link state and destination information between networks in the Internet[57]. Cristallo and Jacquenet[58] first proposed the inclusion of a QoS_NLRI (Network Layer Reachability Information) attribute into the BGP link update messages, advertising the delay, jitter and loss and traffic rate data for a particular link. This extra QoS information can be included in the routing table and is used for the selection of the best route to a destination. QBGP does not provide service differentiation. Its two aims are better resource utilisation and overall better QoS for all traffic. Reference [59] describes an algorithm for egress router selection using QBGP. QBGP is further adopted in the MESCAL[60] project.

2.3.2.6 Agent-Based Approaches

A key feature of agent based systems is autonomous decision making. Within telecommunications, many projects have leveraged this autonomy in an attempt configure resources without direct intervention by a human administrator.

Agent based systems can be divided into two categories: static agents and mobile agents[61]. Static agents, as the name implies are installed in a fixed location and interact with their environment according to their inbuilt behaviour. The distinguishing feature of mobile agents is their ability to temporarily suspend operation and migrate to a new location where operation is resumed. Agent migration is performed because the new location offers some advantage to the migrating agent. Often this is to move closer to a required resource when the size of the new resource is significantly larger than the size of the agent and its attendant resources. For example, a small agent gathering statistics on videos distributed throughout the World Wide Web will prefer to migrate to each video location to analyse the video rather than transfer every entire video across the network. An agent can only migrate provided that the target platform has sufficient storage and processing capacity to accommodate the agent.

Agent mobility is well suited to, and effective within a trusted environment. A largely unresolved problem with agent mobility, however, is with operation in an environment of untrusted and possibly hostile hosts[62][63]. This problem, among others has limited the implementation of mobile agent systems in production environments: especially within the untrusted Internet.

Both fixed and mobile agents may distribute information through message passing. Messages may be sent as part of an interaction protocol, or as solitary utterances. Mobile agents, with the ability to migrate through a network have further options for communication as described below:

An area of research for the application of mobile agents in telecommunications is in the "social insect" paradigm. Swarms of lightweight, mobile agents (often referred to as "ants") are introduced into a network. The ants communicate through a process called "stigmergy" in which small information bundles are left at visited nodes in the network for use by other ants. This information optionally decays over time. In [64] it is argued stigmergy offers a robust process for distributing information throughout a network which can be used to make constraint based routing decisions.

An *Intelligent Network* is **an overlay network for telecommunication systems that facilitates greater configuration sophistication and complexity than the underlying network.** In reference [65], a distributed multi agent system is used for Load Control within Intelligent Networks. **Real-time network configuration is performed in terms of network traffic control, driven by supply-demand economics where underutilised paths are eventually chosen due to the lower demand and therefore price. References** [66] and [8] additionally present agent based systems addressing aspects of network configuration in terms of traffic control for constraint based routing. These works use market based resource allocation and are discussed further in §2.4.2.

References [67][68] and [69] present a fixed agent platform for low level control of ATM networks, addressing admission control and the control of ATM label distribution by intelligent agents. Requests for network resources are directed to resource agents who decide if the request is to be admitted. On success, a resource agent requests one or more Switch Wrapper Agents to configure the network. Each ATM switch in a network is encapsulated by a Switch Wrapper Agent which interacts directly with the switch management interface. Management of ATM paths by intelligent agents is also discussed in [70], in which a two-tier approach is taken. A group of lightweight agents monitor and control the network devices and a smaller group of complex agents make decisions on bandwidth management and virtual path configuration.

2.3.2.7 Active Networks

An *Active Network* (AN) is a telecommunication network in which one or more devices in the network are capable of both forwarding packets as usual, as well as loading and executing mobile code[71]. The code is transported within special packets called "capsules", which may contain the code itself, or a reference to it, to be delivered in later capsules.

The goal of active networks is to provide a set of standard, open network programming interfaces, capable of providing ad-hoc network functionality when used to execute mobile code. An operational AN may be configured on demand to deliver arbitrary services with precise quality characteristics. The roots of AN is certainly in the mobile agent community, originating in the MØ platform[72]. As such, AN suffers from many of the same security problems as mobile agents.

2.4 Resource Allocation

"VoIP is old news. Long live SoIP."[73]

Within telecommunications, there is a drive towards aggregation of independent services and their supporting networks onto a common IP infrastructure. Additionally, new services are continually created and deployed on that same infrastructure. The aggregation of these multiple heterogeneous and quality differentiated telecommunication based services onto a common infrastructure brings many benefits and challenges.

A primary benefit realised is the reduced overhead for operating and upgrading a single support infrastructure compared with multiple dedicated service infrastructures. Utilisation (and revenue) of a network is also increased by allowing low priority services to fill the unused capacity of reservations made for high priority traffic. A major challenge for service aggregation is the fair distribution of the quality controlled resources amongst users.

Fankhauser et al. in [9] state quite simply the problem of service differentiation without appropriate incentives for fair use.

...users who pay flat-rate or low connection time charges will always try to get a better or the best QoS. Users' behavior will be purely greedy, as human beings are always interested in getting the most.

Service differentiation requires incentives for users to choose the service quality that is most appropriate for their needs. It is desirable that data from a low priority email should not compete for resources with a high priority video conference. Additionally, an aim of resource allocation mechanisms is to use the available resources to the maximum efficiency possible.

2.4.1 Allocation Strategies

Most packet-based telecommunication services offer a variable load to a network. Three broad strategies may be adopted to allocate resources given a variable traffic load: Bulk Allocation, Exclusive Use and Over-Allocation.

2.4.1.1 Bulk Allocation

A simple way to address the carriage of different services offering a variable traffic load to a network is to not allocate resources to individual services. Services are provisioned to bulk resources rather than making any individual guarantees. That is, a non-allocation strategy results in "best effort" service delivery. To provide an acceptable level of quality to carried services while using a "best effort" approach to delivery, network resources using bulk allocation are over provisioned as discussed previously in §2.3.1.1.

The vast majority of the current Internet operates using over provisioning with a bulk allocation mechanism. Given that Voice over IP (VoIP), a service with particularly modest resource demands is adversely affected when network utilisation reaches just 17 percent[74], it is not surprising that to ensure an adequate level of service, Internet backbone links typically operate below 15 percent utilisation[5].

A bulk allocation strategy offers no facility to support differentiated services and offers no quality assurance for services. It is highly inefficient in terms of resource utilisation because it attempts to offer a general level of service appropriate to typical high bandwidth applications. Consequently, a large fraction of the provisioned resource designed to not be utilised.

2.4.1.2 Exclusive Use

The *Exclusive Use* allocation strategy involves reserving a block of resources at the maximum level required for the service. This block is reserved for exclusive use and cannot be utilised by other services even if it is not being used.

This approach is used by circuit switched communications and is currently also used by packet switched networks such as ATM and Frame Relay. An exclusive use strategy provides a solid guarantee of resource availability because the entire block of allocated resource is provisioned on the network. ISDN, Time Division Multiplexing (TDM) and Wave Division Multiplexing (WDM) all use an exclusive allocation strategy. ISDN exclusively reserves the required channels from source to destination, TDM reserves one or more time slices and WDM reserves one or more wavelengths.

The hard, deterministic guarantees provided by this approach are obtained at the cost of efficiency. This strategy can be inefficient because bandwidth is a function of time and hence not a storable asset. If the entire allocation is not always utilised, the remaining

5 Real time utilization measurements are available from many sources, http://www.hea.net/mrtg/ (HEANet), http://stryper.uits.iu.edu/abilene/nodeview.cgi?city=losa (Inet2), http://www.aarnet.edu.au/network/mrtg.html (AARNet)

unused portion is simply wasted!

2.4.1.3 Differentiated Over-Allocation

If hard guarantees are not required, and traffic can be allocated with differing certainties placed on expected performance, a more efficient resource allocation scheme may be adopted.

In a *Differentiated Over-Allocation* strategy, more resources are allocated to users than are provisioned on the network, and a traffic differentiation mechanism is applied, allowing differing probabilities for the actual resource availability vs the requested allocation. For new allocations, both existing resource allocations and actual resource utilisation are considered. The case where all resources are allocated with probabilistic availability of 1 generalises to an exclusive use strategy. The goal of an over-allocated resource allocation mechanism is to maintain a near optimal resource utilisation through the widest range of outcome certainties. Figure 10 outlines this concept and compares it with the two alternative strategies.

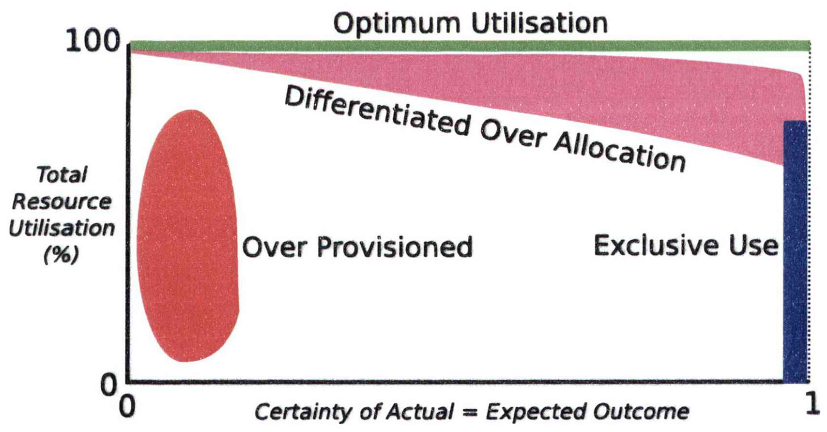


Figure 10: Typical resource utilisation profiles for each allocation strategy.

2.4.2 Allocation Mechanisms

The provision of services has an associated cost. Where a service is provided for financial gain, these costs are generally expressed as a monetary value. If the service is provided with an alternative motive, such as a shared benefit, the costs may also be expressed in financial units, but may be considered in other forms such as time spent or loss of available resource. The FON[4] shared wireless project operates with such a motive.

Typically, service providers desire financial remuneration for the provision of a service. In addition, most telecommunication based services operate in an environment with limited supply and variable demand; a mechanism is required to distribute resources fairly amongst self-interested consumers. An appropriate resource allocation mechanism should address both the provider's remuneration and fair distribution problems.

Various methods are available to allocate resources in situations of limited supply. Where demand is less than or equal to supply, all requests can simply be allocated at the requested levels. However, when demand exceeds supply, the appropriate technique for fair allocation is not clear. A provider may choose to allocate available resources equally amongst requests, irrespective of requested amount. Alternatively, a provider may fulfil complete requests in some order until all the resources are allocated, with some requests potentially refused. A further alternative is to divide the available resources amongst requests in proportion to the requested amounts.

A fair strategy may be seen as one that allocates resources based on actual consumer need. Such a strategy requires an incentive mechanism for the consumer to request resources at a level appropriate to his or her needs. A financial cost corresponding to the quantity and quality of resource allocation gives consumers such an incentive, as a consumer will not wish to pay for unused resources. In a competitive consumer environment (ie. When demand > supply), requests for resources may be accompanied with a willingness to pay, seen as the level at which a consumer values the requested resource. This willingness to pay may be used to allocate the limited resources amongst competitive consumers. Unfortunately, resource valuation based on *willingness* to pay does not accommodate the *ability* to pay. The relationship between willingness to pay (offered price) and true value or need of the resource to the user remains unclear. This disparity may lead to suboptimal resource allocation from the perspective of global satisfaction. Despite this shortcoming, monetary cost has remained a dominant approach to resource allocation for limited[75][76][77] and unlimited [78] resources.

Within the telecommunications domain, numerous approaches to resource allocation have been made. Each is appropriate to a specific problem. Following is a survey of resource allocation mechanisms used in telecommunications. The most generic of these mechanisms; that is, those that may be applied to the widest range of services, are generally based on market mechanisms where currency is used to describe a common value.

Yamamoto and Leduc in [10] discuss a novel mechanism to trade processing power, memory and bandwidth within nodes of an active network, thereby utilizing a network's entire resources more fully. The nodes of the discussed active network each operate a mini-market for available resources and users can trade one type of resource for another as utilization and corresponding market demand change over time. An example implementation is outlined where the desired resources for a real-time audio mixer: storage buffer, bandwidth and computational compression are traded within an active network to deliver the lowest cost service satisfying a stated set of requirements. The work effectively applies market mechanisms to a multi-resource allocation problem, however it suffers from the typical problems associated with mobile agents discussed in section 2.3.2.6.

A platform for negotiating time-based allocations for grid computing services is outlined in [79] and [80], in which the concept of under-constrained requests is introduced. An *under-constrained request* is one in which the request space defined by the constraints is broader than the requirements of the request. In the described implementation, the requirements and constraints correspondingly are: A required duration of service, and an acceptable time window within which the service must be performed. It is shown that the use of such loose allocations enable the service provider to utilise its resources more efficiently than strict reservations.

In [65] and [81], a price-oriented (variable price, fixed resource) and a resource-oriented (variable resource for a fixed price) approach to network load control are compared, finding a price-oriented approach to be more efficient with better convergent properties.

Merkato^{TM}[8] is an agent platform for market-based bandwidth trading and allocation. Merkato uses a Progressive Second Price (PSP) auction[82] which is a decentralised mechanism for allocating variable sized shares of a resource to multiple bidders. The PSP mechanism applies to generic, arbitrarily divisible and additive resources such as network bandwidth. In addition to the direct resource auction mechanism, the Merkato system includes a market for Hold options on bandwidth. These Hold options allow a trader to purchase the right to buy a bandwidth allocation for a future period in order to guarantee allocation at a known price. The restriction of this platform to arbitrarily divisible resources limits its applicability in a operational environment.

In [66], dynamic path determination for real time call routing in telecommunication networks is discussed. Economically rational agents are employed in two different open markets. A localised market for individual links and a path market for end to end paths across a network. An agent is allocated to every link and each candidate path across a network. Path agents buy link resources in the link markets and sell them in the path markets. Additionally, a call agent is deployed at the call source as required to purchase paths from the path markets. Path agents perform the role of market speculators, maintaining an inventory level over links so calls can be placed when required. It is shown that an emergent property of such an approach is an efficient decentralised routing, resulting in the allocation of calls to the best routes when possible, whilst continuing to allocate calls to less attractive paths as resource utilisation increases.

Within the smart market system[83] bandwidth within a single service network is allocated at the packet level. Each packet has an associated bid and packets are served in order of their descending bid prices. All packets served pay the price of the cutoff bid. That is, the highest bid of the packets that were denied service. This mechanism is incentive compatible and truth telling in a competitive market. Although multiple service classes are not explicitly offered, in practice, the price ordered servicing of packets creates a traffic prioritisation mechanism with fine granularity. A disadvantage of such an approach is the extra processing of packets required to service them in price order. Such overhead may be unacceptable on a high bandwidth link.

References [84][85][86] and [87][88] all employ resource pricing and agent frameworks to allocate limited telecommunication network resources within a competitive environment. It can be seen that such an approach to resource allocation is a fertile research area with a variety of automated allocation techniques available.

In addition to the automated approaches, traditional bandwidth markets exist for long-term network resources [89][90][91]. Although highly manual in nature, these markets demonstrate the operational use of market mechanisms for resource allocation.

The use of market mechanisms and differentiated pricing for resource allocation not only addresses the problem of fair distribution, but also of provider remuneration. The level of remuneration is determined by the actual prices applied to the services.

2.5 Service Pricing and Charging

Service Pricing involves determining a price for a given service. Assuming self-interested players, both providers and consumers aim to get the best or "most preferred" price/service combination. It is clear that the preferred price for the consumer is a low price, and the preferred price for a provider is a high price.

A service has a set of parameters that describe its functional and non-functional characteristics. Each parameter may affect the perceived value of the service to a player. When comparing two services, it is clear that if service "A" has a preferred value for every parameter taken individually over service "B", then service "A" is preferable to "B". If some parameters on a service are preferred and others not, the preference at that level of service is not so clear. Additionally, the preference of one service configuration over another cannot be assumed to be consistent amongst players.

In section 2.4.2, it is established that the price a consumer is willing to pay for a resource can be determined by how much they value the resource. Given a service with multiple parameters, a method is needed to determine the relative value of different configurations of these parameters. Preference elicitation and value modelling offer an approach to handle this complexity.

Preference elicitation is the process of extracting the necessary preference or utility information from a user[92]. It is largely concerned with human computer interaction. *Value modelling* deals with the problem of modelling user preferences in a formal, efficient and meaningful way, allowing operations to be performed on those preferences.

Given an acceptable model of consumer preferences, an attempt at pricing service resources can be made. Service pricing is composed of two components:

1. A *Charging Mechanism* that defines the relationship between outgoing resources such as bandwidth and incoming resources; typically money for providing a service. The charging mechanism defines how charges are attributed to a consumer for the use of a service. An example is hourly charging. A charging mechanism has significant impact on how a consumer uses a service[93].
2. A *Pricing Algorithm* that determines the revenue received from the provision of a service, for example $10 per hour. Often, the pricing algorithm is directly related to the resources consumed by the service.

Service pricing is composed of technical concerns and market aspects. Technical concerns typically favour a cost pricing approach (How much does the service cost to deliver?). Market-based pricing explores the value the market as a whole places on the service, and hence what consumers are prepared to pay. A further pricing concern is whether the price of a service is set at the start of a service and fixed for the life of the service, or if the price varies during the life of the service. Some charging mechanisms are suited to fixed pricing, others to variable, while many are applicable to either approach.

The range of charging mechanisms can be divided into a set of categories. These categories are not mutually exclusive, and a final charging mechanism may use techniques from multiple categories.

2.5.1 Flat rate

Flat rate charging refers to a tariff that is independent of the amount of resources used or grade of service. A flat rate is usually charged at a predetermined regular interval, but may be a one off charge. With a flat rate charging scheme, the consumer is effectively paying for a set service[94]. Most local telephone services in the USA are based on a flat rate charging scheme. A benefit of flat rate pricing is that providers do not need to track usage and it is easy to set prices and bill for services. However, based on this scheme, low-load users effectively subsidise high-load users.

2.5.2 Expected Capacity

The *Expected Capacity* mechanism uses the expectation a consumer has of potential use [95] as a basis for charging. Consumers nominate their expected peak resource load and are charged accordingly. The resource level at the expected capacity is made available from the provider. This mechanism allows differentiation between low and high load users. As McKnight and Bailey outline in [42], Expected Capacity is the prevailing mechanism used for charging on the Internet. Expected Capacity charging requires more complex service configuration and billing systems, however the complexity of usage tracking is not required.

2.5.3 Usage Based

When *Usage Based Charging* is adopted, the resource consumer pays only for the resources used. The provider of a usage charged service must record the actual usage of its customers. Usage based charging therefore requires a larger administrative effort from the provider. Additionally, the consumer has less consistent payments. Research has suggested that some Internet users do not react favourably to usage based charging schemes[93][42]. Examples of usage based charging are found in the time base charging model used for mobile telephone calls, or a cost per megabyte for an online backup solution.

2.5.4 QoS Sensitive

The QoS Sensitive charging mechanism applies charges according to the level of service quality requested. A consumer nominates a particular quality when requesting the service and is charged for that quality. QoS Sensitive charging is similar in concept to expected capacity charging, although QoS Sensitive charging is not what a consumer *expects* they will use, it is *exactly* what they want and expect to receive. A service that supports the delivery of multiple different qualities of service will likely use some form of QoS charging.

2.5.5 Congestion-Based

With *Congestion-Based Charging* mechanisms, charges are applied, dependant on the combined utilisation of a shared resource. Peter Key in [96] discusses the merits of congestion charging. The common assumption that resource reservation or call admission control is necessary for guaranteed services is challenged. Key argues that if the network can set the correct price, based on reasonable "price matching" of users to the network, then users will decide whether to admit themselves or not, and strict network enforced controls are not required. This is one of the principal motives behind congestion-based charging.

A further example of congestion-based charging is the work performed by MacKie-Mason et. al. in [97]. The concept of smart market pricing is introduced, where charges are accrued from a continually clearing Vickrey auction. Packets are admitted if their bids exceed the cut-off amount, determined by the marginal congestion costs imposed by an additional packet. The Smart Market mechanism is described further in §2.4.2.

2.6 Ubiquity

Today's telecommunication environment has a wide range of service providers, operating a vast range of networks and services. At any point in space and time, a service user may have physical access to a number of fixed and mobile networks, administered by different entities and using technologies such as Ethernet, WiFi, WiMax, GSM, UMTS, ADSL or Bluetooth. All telecommunication-based services are basically concerned with the transfer of data from source to destination, while the way in which that data is transferred varies. As such, some service are more suited to some network types, while other services could potentially operate over any available network, providing the desired destination can be reached.

In current telecommunications, the delivery of a particular service is typically bound to a particular type of network technology and/or administrative body. Often, this is to ensure an appropriate level of quality for a service, or because appropriate business relationships do not exist. In new generation networks, it should be possible to deliver quality assured services over any available network that is technically appropriate; independent of the particular administrator. In a *ubiquitous* service environment, services are managed over technologically and administratively heterogeneous delivery infrastructures. Two key component of such an environment are: managing the technical handover of an active service between heterogeneous delivery domains, and the management of dynamic business relationships required to support the services.

2.6.1 Mobility

Service ubiquity is primarily driven by user desire for mobility. The desire for mobility is introduced in sections 1.1 and 2.1. A ubiquitous environment offers enhanced mobility by extending the delivery of a mobile service to all available technologies that can potentially support the service. Service ubiquity within a telecommunications environment may be achieved by addressing the technological and administrative concerns of the following four aspects of mobility:

1. Terminal – The DOLMEN project[98] offers the following definition for terminal mobility: "Terminal mobility enables end-users to access and use services through mobile terminals, and to maintain service provisioning while roaming within a suitable geographical area." An example of terminal mobility is the technical handover that currently occurs as a user travels between cells of a mobile network.

2. Session – The TINA Consortium architecture document[99] states that "Session mobility allows a user that has an active session on a particular terminal to move that session to another terminal". An example of session mobility is the transfer of a video-on-demand service session from a home theatre pc onto a mobile device such as a PDA for continued mobile viewing.

3. Service – Service mobility allows the use of a service to be independent of the provider delivering the service. Additionally, an active service may be transferred from one provider to another. As an example, consider the scenario a VoD service is being provided to a user by some provider, and the user moves into a different provider's domain, the service should be transferred to the new provider; ideally without interruption.

4. Personal – Personal Mobility is defined by the TINA Consortium in [99]: "Personal mobility enables users to use services that are personalized with their preferences and identity ubiquitously, independently of both physical location and specific equipment". Additionally, the DOLMEN project defines personal mobility as enabling "endusers to access and use services worldwide independently of time, place and the terminal being used"[98]. A weak version of personal mobility is delivered through the use of mobile phone SIM cards. A user may put his or her SIM card in any accepting mobile phone and have his or her personalised services available on that phone. SIM cards however, are currently only used for services from a single provider.

In this section, the place of ubiquity within new generation networks is discussed, along with the way it may be achieved through addressing four aspects of mobility. We follow with a critique of the current art in SLA Specification and negotiation platforms.

2.7 How can we describe the services?

Throughout modern history, as business expanded, greater reliance has been placed on the formalisation of deals. As business size, internationalisation, scope and interdependence have increased, formal written contracts have replaced spoken word, increasing the level of accountability and providing strict guidelines on the goods and services to be provided. In a service industry, these contracts are termed *Service Level Agreements* (SLA).

SLAs have become the de facto agreements for the service sector. In the IT domain particularly, where outsourcing is the norm, these "e-business" contracts have become pivotal to conducting business.

The need for automatically negotiated SLAs in the telecommunications environment is driven by future projections concerning the delivery of complex new generation services, and by the need to reconfigure services quickly in response to changes in requirements. These agreements specify guaranteed and differentiated levels of quality for services delivered over a telecommunications infrastructure.

As the notion of 'quality of delivery' becomes better understood, the delivery of these services should become dynamic and have a fine grain of control to ensure efficient use of service resources. The SLAs should be comprehensive and accurately express the quality of delivery.

Concepts expressed on an SLA should have a common understanding amongst involved entities. Achieving a common understanding across a large, heterogeneous environment such as the Internet, requires an abstract formalism for representation and a standard syntax. Research has identified the need for a common SLA format with semantic annotations[100].

The CADENUS project [101] describes SLAs as a high level of abstraction for specifying service requirements within a Service Level Management (SLM) framework. Such a level of abstraction and subsequent implementation independence renders SLAs an appropriate format for communicating service requirements between heterogeneous SLM implementations.

Much work has been done in the field of syntactic interoperability on the World Wide Web. A popular solution is through the use of XML(§2.7.1). Authors have claimed that

XML is a "silver bullet" for future e-business integration and automation[102]. Others have shown this is only partially true, and for effective integration, a semantically richer framework such as RDF(§2.7.1) is required[103].

The use of XML and RDF effectively deals with the requirements of a common syntactic framework for SLAs; however RDF only goes part of the way towards expressing the semantics or "understanding" of an SLA.

Previous research has outlined the main drivers to the specification of ontologies for a specific domain [104]:

- To share a common understanding of the structure of information among people or software agents.
- To enable reuse of domain knowledge.
- To make domain assumptions explicit.
- To separate domain knowledge from the operational knowledge.
- To analyse domain knowledge

These drivers align strongly with many requirements for an SLA representation. The use of a suitable ontological language, along with a common syntactical framework may therefore be employed to address the requirements of SLA formalisation in a heterogeneous telecommunications environment.

2.7.1 The Semantic Web

The *Semantic Web* aims to establish a distributed network of relations between data specified in XML. All semantic web technologies are based on The *eXtensible Markup Language* (XML)[105] and add semantic strength in layers through the addition of XML tags which have specific meaning when evaluated at a specific layer of the semantic stack (Figure 11).

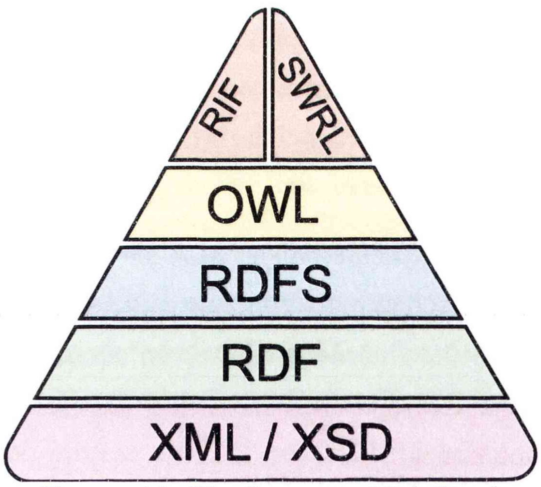


Figure 11: The Semantic Web Stack

XML is a World Wide Web Consortium(W3C) standard data representation language that allows the definition of meta tags, used to further describe a block of data. A tagged block of XML data is defined through enclosure between opening and closing named tags. Example 1 Illustrates an XML block, where it can be seen that gearbox is a component of car

```
<car>
   <gearbox>Automatic</gearbox>
</car>
```

Example 1: An XML block

because it occurs between the opening and closing `<car>` tags. An *XML Schema Definition* (XSD) may also be used to describe the correct structure of an XML document, an XML document can be verified as conforming to a particular structure as defined in an XSD. The XML Schema in Example 2 describes a car as being composed of two elements: a gearbox and a number of doors. Both these elements are optional.

```
<xs:element name="car">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="gearbox" type="xs:string"/>
      <xs:element name="doors" type="xs:integer"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Example 2: XML Schema for Example 1

The combination of XML and XML Schemas allows the description of a limited number of predefined relationships between XML objects, such as the "has-a" relationships (eg. A car has a gearbox) and an extension relationship (eg. An Automatic car is an extension of a car, defining an automatic gearbox). These relationships are useful, but cannot express a large range of relationships between elements such as "cousin" or "references".

The *Resource Description Framework* (RDF)[106] introduces the ability to identify XML blocks within a universal scope and define explicit relationships between pairs of tagged blocks. All relationships are binary with an abstract representation of subject:predicate:object triples. A collection of RDF statements intrinsically represents a labelled, directed pseudograph. The W3C standard concrete representation defines nodes of this graph as Resources and labelled edges as Properties. Properties define the relationships between tagged XML blocks (Resources). Example 3 describes an rdf resource named "mycar", identified by the namespace "http://car.com". "mycar" is related to the resource "Automatic", identified by the namespace "http://gearbox.com" through the "gearbox" property, identified by the namespace "http://gearbox.com".

```
<rdf:RDF xmlns="http://car.com" xmlns:gearbox="http:gearbox.com">
  <rdf:Description rdf:ID="mycar">
    <gear:gearbox rdf:resource="http://gearbox.com#Automatic/>
  </rdf:Description>
</rdf:RDF>
```

Example 3: An RDF block

RDF Schema (RDFS)[107] allows the assignment of tags to classes and a defined hierarchy of classes and RDF properties. In RDFS, the range and domain of properties can be restricted to be of specific classes. Both RDF and RDFS are standardised by the

W3C.

The *Web Ontology Language (*OWL) [108] is a W3C standard ontology language designed to work seamlessly with XML, RDF and RDFS and is rapidly gaining support and acceptance in the ontological research and business community. OWL is semantically rich and builds on RDF and RDFS adding more vocabulary for describing properties and classes. OWL makes the following additions to the semantic web stack.

- Enumerated classes and relations between classes (e.g. Disjointedness, inheritance)
- Cardinality restrictions on properties (e.g. "exactly one")
- Object equality
- Richer typing of properties and property characteristics (e.g. Symmetry)

OWL operates under an “open world” assumption, meaning no assumption can be made on anything that cannot be inferred, or put simply; “if it's not stated, it doesn't mean it's not true”. The OWL open world assumption has the following basic rules:

- New information cannot retract previous information.
- New information can be contradictory.
- Facts and entailments can only be *added*, never *deleted*.

Example 4 illustrates the use of an OWL class to create a new RDF resource conforming to that class. The lower section of Example 4 describes a class of RDF resource named Car, and restricts every car individual within that class to be related to exactly one gearbox through the “hasGearbox” property defined in the “http://gearbox.com” namespace. The upper section of Example 4 describes a use of the Car class.

```
<rdf:RDF xmlns="http://example.com"
         xmlns:car="http://car.com"
         xmlns:gear="http://gearbox.com">
  <car:Car rdf:ID="mycar">
    <gear:hasGearbox rdf:resource="http://gearbox.com#Automatic/>
  </car:Car>
</rdf:RDF>
----------------------------------------------------------------------

<!-- extract from car.com definition file -->
<rdf:RDF xmlns="http://car.com">
  <owl:Class rdf:ID="Car">
    <owl:subclassof>
      <owl:Restriction>
        <owl:onProperty rdf:resource="http://gearbox.com#hasGearbox">
        <owl:cardinality>1</owl:cardinality>
      </owl:Restriction>
    </owl:subclassof>
  </owl:Class>
</rdf:RDF>
```

Example 4: An OWL description

OWL as a language has three sub languages. OWL Lite, OWL DL and OWL Full. OWL Full allows maximum expressiveness, but queries over the ontologies built may be undecidable. OWL Lite and DL both maintain computational completeness and decidability, meaning all conclusions are guaranteed to be computable and will finish in finite time[109]. OWL Lite is primarily for building classification hierarchies with simple constraints. OWL DL allows automated reasoning whilst remaining strongly expressive. Finally, OWL is a self documenting language. Metadata can be applied to classes, properties or entire ontologies describing further information for these components.

Rules, constraints and relationships need to be maintained within ontologies to ensure complete and correct knowledge representation. OWL alone cannot sufficiently express a proportion of these rules, so some additional rule language is required. XML based languages to describe rules and first order logic exist, however none have achieved W3C standardisation. The *Semantic Web Rule Language* (SWRL)[110] has been an active submission to the W3C and is popular amongst the research community.

SWRL expresses rules as function free horn clauses. SWRL has an abstract syntax in addition to a concrete syntax for use with OWL. In 2005, the W3C announced the Rule Interchange Format (RIF)[111] Working Group. The group aim is to bring together web based rule technologies and develop a standard rule language for the semantic web.

2.7.2 Modelling Ontologies

Many languages are available for modelling machine comprehensible ontologies. OWL[108], DAML+OIL[112], OCML[113], KIF[114], Cyc[115] and Loom[116] are some examples. Two leading approaches are KIF and OWL. DAML+OIL was used for the basis of OWL development, Loom is based on a variation of KIF and OCML is not widely used.

The *Knowledge Interchange Format* (KIF) is a logic-based, computer-oriented language for the exchange of knowledge. It has declarative semantics; it is logically comprehensive (i.e. it provides for the expression of arbitrary sentences in the first-order predicate calculus); it provides meta modelling, that is, it can represent knowledge about the representation of knowledge; it provides for the representation of non-monotonic reasoning rules; and it provides for the definition of objects, functions, and relations. KIF originated as a LISP application and the concrete implementation is expressed in LISP. KIF allows ontologies to be undecidable and therefore general methods for reasoning over those ontologies are difficult to specify. Re-use of existing ontologies, and the ambiguities that can arise by doing so are not addressed directly in KIF.

The Web Ontology Language (OWL) is a frame-based knowledge representation language for ontology formalisation on the web. OWL does not allow full first-order Logic expressions, but expresses knowledge through binary relations between individuals. OWL also provides meta modelling. The concrete implementation of OWL is based on semantic web technologies and as such allows unique identification and re-use of ontology concepts.

The information model used by OWL conforms to the Meta Object Facility (MOF) layered approach(Figure 18) defined by the Object Management Group[117]. The MOF is a four layer information model consisting of:

- The information layer (Layer 0) that comprises the data to be described.
- The model layer (Layer 1) that comprises the meta-data that describes data in the information layer.
- The meta-model layer (Layer 2) that comprises the descriptions that define the structure and semantics of the meta-data.
- The meta-meta-model layer (Layer 3) that comprises the description of the structure and semantics of meta-meta-data.

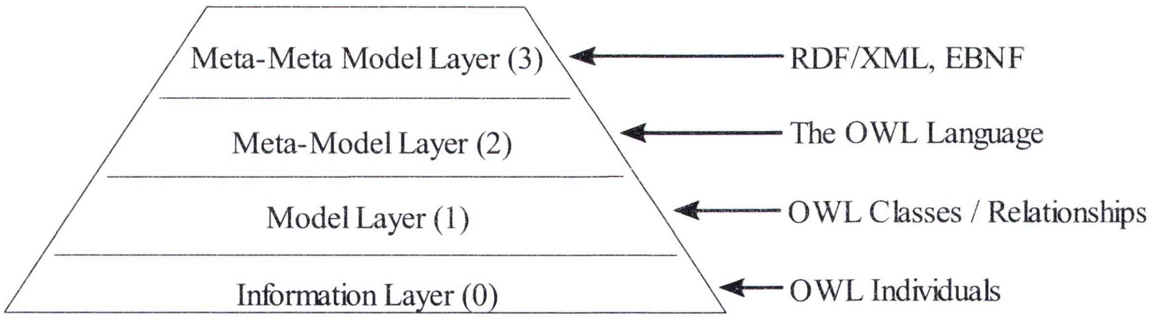


Figure 12: The MOF metadata stack

Following the MOF model, final OWL individuals belong to the information layer; the classes, relationships and rules specified in OWL are at the model layer; the OWL language itself is the meta-model layer; and the language in which OWL is defined is the meta-meta-model. The OWL language is described in both an abstract syntax using Extended Backus-Naur Form (EBNF)[118] notation to aid human understanding, and a concrete syntax expressed in RDF/XML[119] for use on implementation within a functioning system.

2.7.2.1 OWL

Within OWL, there are three main types of resource: Classes, Properties and

Individuals. Classes can be defined via any of the following 6 mechanisms[120]:

1. a class identifier (a URI reference)
2. an exhaustive enumeration of individuals that together form the instances of a class
3. a property restriction – Defines a class of individuals by the conformance of those individuals' properties to a set of property restrictions.[6]
4. the intersection of two or more class descriptions
5. the union of two or more class descriptions
6. the complement of a class description

Individuals can be created with multiple types. For example, if there are two classes: `PoliceMan` and `Father`, an individual created from the `PoliceMan` class can also be assigned to the class `Father`. Such an individual will have all the properties of both `Father` and `PoliceMan`. An important point here is that an `allValuesFrom` property restriction with the range of `Father` will remain consistent even if one of the `Father` individuals is also specified as belonging to another class. The `allValues-From` restriction does *not* mean that an individual used in the range of that property must belong *only* to the restricted class, but every individual used must *at least* belong to that class, but may additionally belong to other classes.

There are a set of in-built relationships available from OWL, RDFS and RDF for use with OWL Classes, Properties and Individuals. Some examples are: `subClassOf`, `inverseOf`, `sameAs` and `equivalentClass`. In addition to these pre-defined, in-built relationships, an OWL `Property` is used to specify relationships between OWL individuals or between individuals and literal data types. A property is a uni-directional relationship from an individual indicated in the domain of the property to a concept indicated in the property's range. A relationship between two individuals is an `ObjectProperty`, and a relationship between an individual and a data literal is a `DatatypeProperty`. The domain of any property can be constrained to one or more classes, as can the range of Object Properties. The range of data-type properties can be constrained to any of the XML Schema data types.

Every OWL ontology has a globally unique namespace defined by its Uniform Resource Identifier (URI). Every named concept within an OWL ontology can be identified within a global scope by the concatenation of its containing ontology

6 A property restriction is also used to restrict the cardinality of, or values allowed in the range of a property when used in individuals of some class.

namespace with its concept identifier. Any OWL ontology may import another OWL ontology, making all the concepts from the imported ontology directly available to the importer. The namespaces of the imported concepts do not change when their ontology is imported.

2.7.2.2 SWRL

To ensure a complete and correct knowledge representation; rules, constraints and relationships are maintained within ontologies. The *Semantic Web Rule Language* (SWRL)[110] has been defined to address the requirement of rule expression within OWL ontologies. SWRL is an active submission to the W3C and is popular amongst the research community. Within SWRL, rules are expressed as Horn clauses, and an abstract syntax is defined in addition to a concrete syntax for use with OWL. There are numerous reasoners available for the SWRL language such as Bossam[121] and Hoolet[122].

SWRL is is based on RuleML[123] and there are three concrete syntaxes for describing SWRL rules; XML, RDF and OWL. All three concrete syntaxes allow a different level of expression. The XML syntax is most generic as it can define rules over any XML component. The OWL syntax is most specific, where components of rules are limited to specific OWL concepts.

With the OWL syntax, rules are represented by the class `Imp`. Every `Imp` has a `head` and a `body` that respectively represent the consequent and antecedent in a Horn clause. Both the `head` and the `body` of an `Imp` are lists of predicates. Each predicate is represented by the parent class `atom`, and holds over one or more arguments that are either unbound variables or existing OWL individuals. Atoms used within rules are required to be one of three sub-classes: `ClassAtom`, `IndividualPropertyAtom` or `BuiltinAtom`.

- A `ClassAtom` is satisfied when the indicated argument is of a stated OWL class.
- An `IndividualPropertyAtom` states a relationship between two indicated arguments.
- A `BuiltinAtom` defines a custom operation over one or more arguments and has a Boolean result.

OWL knowledge bases containing arbitrary SWRL expressions may be undecidable for certain inference problems[124]. A motivation for selecting OWL-DL as the ontology description language is that knowledge created within OWL-DL ontologies is guaranteed to be decidable. An identified subset of SWRL rules are "DL-safe"[125], meaning that they maintain decidability of the ontologies over which they are used. An SWRL rule can be made DL-safe when each variable that occurs within is tied to a specific OWL individual in the rule body.

SWRL rules are function-free Horn clauses. As such, all variables occurring in the rule consequents are universally quantified. SWRL has no native mechanism to explicitly quantify variables. Existential quantification can only be expressed with the use of OWL-DL constructs and even so, some existential constraints can not be expressed. The First-Order Logic (FOL) expression in Formula 5 is required within the ontologies used in this work. Such expressions using existential qualification are not directly supported in SWRL.

$$\begin{gathered}(\exists ?th, ?rf, ?ht).\mathit{OntologyTreeHash}(?th) \land \mathit{HashTree}(?ht) \land \\ \mathit{hasOntologyFile}(?th, ?rf) \land \mathit{hasHashTree}(?th, ?ht) \Rightarrow \\ (\exists ?fh).\mathit{OntologyFileHash}(?fh) \land \mathit{hasOntologyHash}(?ht, ?fh) \land \mathit{hasOntologyFile}(?fh, ?rf)\end{gathered}$$

Formula 5: An impossible rule using pure SWRL

Research has highlighted this weakness and the need for partially quantified constraints in SWRL [126][127]; specifically the existential quantifier. There are two RDF based approaches for enabling existential quantification. Each of the approaches has outlined an integration strategy with OWL and SWRL. Unfortunately, at the time of writing, neither is nearing standardisation.

A novel solution is proposed by Mackenzie, Gray and Preece in [127]. Within that work, a concise RDF vocabulary is proposed, based on the Constraint Interchange Format (CIF) [128]. Utilising and integrating with the existing work done with OWL and SWRL, the CIF syntax adds the ability for existential quantification of variables and nested implications. Alternatively, the proposed RuleML First Order Logic Web Language[129] and SWRL extensions [130] provide the ability to express arbitrary FOL expressions within RuleML and SWRL, including existential quantification of expression variables.

An approach for standardised rule interchange within the semantic web framework is addressed by the recent Rule Interchange Format[111] (RIF) working group. At the time of writing, no specification is available.

2.7.3 Towards an appropriate format

Service Level Agreements contain a combination of functional and non-functional components. Functional components describe the technical parameters of a service, required for operation of the service in a specific way. Some functional parameters are the bandwidth to use for a connection, or a specific video resolution. Non-functional components of an SLA specify additional information that either party may need to enter into an agreement for service provision. Provider and consumer business details, charging information and conditions of service are some of the non-functional components that may be associated with an SLA.

Research on various SLA structures in use throughout the telecommunications industry and similar industries has led to proposed requirements in defining SLAs.

- Security – When a service works with sensitive information it may have specific security requirements.
- Monitoring – Customers may require methods of monitoring a running service to ensure compliance with the stated goals on the SLA. These monitoring methods should be addressed in the SLA structure. The WSLA specification developed by IBM [88] uses a novel approach to specifying monitoring parameters where monitored metrics may be individual or composed using an aggregation function from individual metrics.
- Accounting – Attributes relating to the pricing and charging of a service as well as possible additional billing information may be included. (Eg. Reverse Charges)

Much work has been done towards service level representation. The TEQUILA project [131] has defined a Service Level Specification (SLS) specific to DiffServ networks. Traffic is defined in terms of simplex flows and the SLS defines succinctly the characteristics of such a flow. Technical parameters such as path ingress and egress addresses, token bucket rates, along with flow identifiers such as protocol number, port id, source and destination IP addresses are defined. Guarantees are given for packet delay, packet loss, jitter and throughput. Other non functional SLA components such as pricing are not addressed by an SLS.

The TEQUILA SLS has been extended by MESCAL [60] into a Service Subscription Specification (SSS). The SSS defined in [132] outlines a technique for aggregating uni-directional service specifications into a subscription offering. An SSS includes non functional information about the subscriber, the service schedule, availability and reliability guarantees, in addition to how the services will be started and stopped. Financial information such as service pricing is not included.

Alcatel has made significant progress in specifying the requirements of a Service Level Agreement Management framework [133][134]. Requirements identified for Service Level Agreements include Customer and Service Provider responsibilities, SLA violation procedures, service description, customer reporting and configuration by the customer.

The Euroscom project P1008 explores the management of IP connectivity services with a particular focus placed on DiffServ networks. The final technical specification document[135] divides the components of service management through the use of individual information blocks; an InterConnect Agreement (ICA) block, Traffic Forecast (TF) block and SLS / Service Description (SLS/SD) block. The ICA block is concerned with non-functional information related to business relationships. The SLS/SD block describes functional and non functional information related to a particular service. Concepts such as scope, topology, traffic identification validity, reliability and monitoring are addressed. The SLS/SD block is similar to the TEQUILA SLS, with the addition of non-functional information. Finally, the TF block enables the consumer to describe its predicted service usage. The approach taken by the P1008 project offers a comprehensive suite of attributes for describing an instance of a IP connectivity service, however the framework is very specific to IP connectivity and unsuited for use with a variety of service types. SLAs are described in UML with discussion given to a possibility of implementation in XML.

In [136], Service Level Agreements are represented as a set of commitments. These commitments describe actions, obligations and changes of contract state from the occurrence of events. A contract tracking ontology is formalised in XML. Modification to contracts is caused by events and based on an Event Calculus[137]. The main contribution of [136] is in defining the relationship between events and changes in the contract state. Functional SLA parameters are briefly discussed.

The SLA language (SLAng)[138] defines an XML syntax for describing Service Level Agreements. The goal of SLAng is to provide a generic platform, able to describe SLAs from many service domains such as Internet Service Provision, Application Service Provision and Storage Service Provision. The importance of formal semantics attached to an SLA is recognised, and these semantics are expressed through an abstract UML description of SLA components, along with natural language and use of the Object Constraint Language (OCL)[139]. No separation is made between functional and non functional SLA parameters. The notion of inter-service composition is raised. Inter-service composition explores SLA compatibility in terms of satisfaction. An SLA satisfies another when the constraints on one SLA are stronger than another. The ability to reason over created SLAs is not addressed.

Rule Based Service Level Agreements have been proposed by Adrian Paschke in [140].

Service Descriptions are augmented by a set of Event-Condition-Action (ECA) rules defining guarantees applicable to a service. These rules are specified using an extension of the Rule Markup Language (Rule-ML). This approach is similar to [136] and focuses on the impact on an agreement due to changes in the physical world rather than the definition of generic SLA concepts.

2.7.3.1 Web Services

Web Services are software applications that are made available through a standardised interface as a programming language and operating system independent service available over the web. Originally proposed as a technical solution to software interoperability, Web Services forged the concept of Service Oriented computing. Economic impacts of a service oriented approach have since been considered, and attempts to define service agreements around the web service concept have emerged.

IBM made significant progress on the specification of Service Level Agreements for Web Services (WSLA)[88][141]. The WSLA project defines an XML Schema for Service Level Agreements for Web Services, along with a distributed monitoring framework to track compliance with the agreements. A WSLA is composed of a Parties section, Service Definition section and a set of Service Obligations. A Service Definition includes one or more Service Operations describing a particular functionality that a provider supplies. Parameters used in this operation describe an observable property of the service whose value can be obtained from a source of measurement. A Parameter has one Metric assigned to it, which defines how the value of the Parameter is computed. This Metric is used to determine the Parameter's value at execution time. Finally, the Service Obligations section defines a set of service level objectives defining the required state of parameters, along with a set of action guarantees defining promised actions by obliged parties.

The work performed by HP Labs [142] towards SLAs for Web Services is also significant. An SLA as expressed in [142] consists of an activation period and one or more Service Level Objectives (SLO). Each SLO has a time it is active within a day and a set of clauses. Clauses provide the details on expected performance and define what is being measured, when its being measured, what the performance evaluation function is, and when that function is to be evaluated. Information about involved entities, service pricing or charging is not addressed.

The Web Services Agreement Specification (WS-Agreement)[143] is a Global Grid Forum (GGF) initiative for defining Service Level Agreements for Web Services.

Extending the WSLA work performed at IBM, an WS-Agreement includes 3 components(Figure 13); Agreement name, context and terms. The agreement context defines the parties to the agreement, expiration time and a template if applicable. The main body of an agreement consists of a set of agreement terms. These agreement terms are either service terms or guarantee terms. Further, service terms are separated into Service Description, Service Reference and Service Property terms. Service Description Terms hold a domain specific description of the delivered service. This may be described in a language independent of the WS-Agreement. Service Property Terms define the functional attributes of a service in terms of a set of variables, these variables are metrics including a location from where the value of the metric may be found. Finally, guarantee terms define the actual constraints and obligations under which services must operate, along with any reward or penalty for (non)compliance. Pricing or charging information is not addressed.

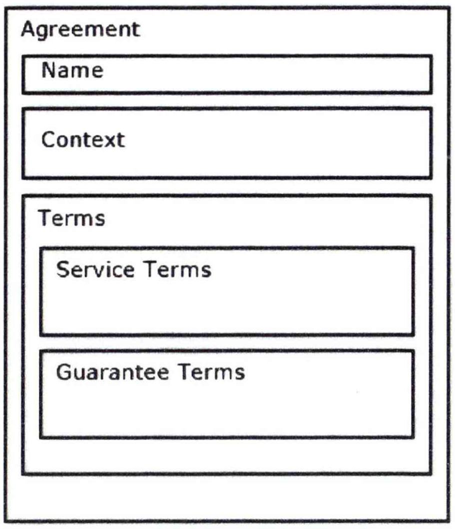


Figure 13: Stucture of a WS-Agreement

2.7.3.1.1 Semantic Web Services

Semantic web services aim to augment the web services architecture with features from the semantic web. OWL-S[144], SWSF[145], WSMO[146] and METEOR-S[147] are recent approaches to developing semantic web services.

OWL-S is an OWL-DL ontology for expressing web service concepts. A service, as described by that ontology, consists of three components. A *Service Profile* that describes what the service does, in other words, how activation of the service will change the world. A *Service Grounding* describes how to access a service within an actual implementation and finally a *Service Model* describes how a service works in terms of the processes involved to use the service. OWL-S deals with describing the web service model using a standard ontology language. No attempt is made at the specification of Service Level Agreements.

The Semantic Web Service Framework (SWSF) defines a First-order Logic Ontology for Web Services (FLOWS). FLOWS captures all the concepts in OWL-S with extensions on the process model of web services. Based on First Order Logic, FLOWS is is more expressive than OWL-S which is limited by the decidability constraint of OWL-DL. FLOWS is defined in abstract form and is *Grounded* for a particular use by mapping the abstract concepts to a concrete syntax using technologies such as WSDL. SWSF describes services semantically, but does not address service level agreements.

With a foundation in the Web Service Management Framework (WSMF)[148], and similar to SWSF, the Web Service Modelling Ontology (WSMO) is a conceptual, abstract approach to Semantic Web Services. The Web Services Modelling Language (WSML) provides a grounding in a concrete syntax, and the Web Service Execution Environment (WSMX) provides an execution architecture and reference implementation. The WSMO approach is from the ground up; it first defines an abstract ontology language to express required concepts, then describes concepts in the domain of web services such as capability and interface. The ability to describe non-functional components of a web service is discussed and the addition of annotations such as a description to service components is briefly mentioned. Service Level Agreements are not included in the framework.

METEOR-S has four main goals for the convergence of semantic web technologies with web services:

- Semantic Annotation and Publication of Web Services
- Abstract Process Creation
- Semantic Discovery of Web Services, and
- Orchestration/Composition of Web Services.

METEOR-S uses a semantic mark-up of the Web Service Definition Language called WSDL-S[149] and integrates the Business Process Execution Language for Web services[150] to provide semantic process description. WSDL-S shares similar motivation as OWL-S but aims to be a more lightweight implementation, extending the existing WSDL through the addition of extra tags by which a semantic description of components can be attached.

As can be seen from the above work, progress on semantic web integration with Web Services is advancing rapidly. Research on defining complete Service Level Agreements for semantic web services is not so advanced. The Official Summary Report from the 2005 W3C Workshop for Semantics in Web Services[151] discusses the need for work on non-functional aspects of web services including electronic contracts and Service Level Agreements. Preliminary work in this area is found in [152] and [153]. These exploratory works map the WS-Agreement XML Schema into OWL ontologies, enabling semantic provider/consumer matching for web services within the WS-Agreement framework.

2.8 Critique of the Current Art

2.8.1 SLA Specifications

SLAs used within existing research often focus purely on a functional service description, and ignore the non-functional information such as pricing or charging which distinguishes an SLA from a service description. Often some knowledge exterior to the SLA specification is required for operation of the services covered on the SLA, negatively impacting reuse of the SLA specification within different operational frameworks.

Many SLA specification techniques are specific to a particular service type, such as in [14],[135] and [154]. In a multi-service environment, one SLA specification language should be able to describe SLAs for multiple service types. An additional problem with current work is that the expression of an SLA is bound to use within a particular communication framework, as in [100] and [154]. That is, there is often no clear separation between the framework and the service descriptions. To promote interoperability, the service specification should stand alone and not be bound to interpretation within a particular framework.

A Service Level Agreement is a legal contract for the delivery of a service. Existing approaches to specifying SLAs typically ignore this characteristic of an SLA, or the security of the SLA is assumed to be delivered by some external technology[85][143]. When developing an SLA specification technique, the potential attacks on the validity of the SLA should be considered from conception.

Effort has been made in specifying SLAs based on web services[141][143], however these approaches generally lack formal semantics. Recent research [144][145] has worked on the addition of semantic annotations to web services. “Semantic Web Services” add formal semantics to the web service framework. However, research in formal semantics within complete Service Level Agreements for web services is embryonic. The limited research in this area [153] has yet to address many requirements for business grade SLAs, such the security issues introduced when using distributed semantics for agreements, or grounding of the agreements to real measurements. Semantic service level agreements are still largely an open research issue.

2.8.2 Negotiation Frameworks

Much research in service negotiation mentions SLA negotiation, however the vast majority focus purely on the technical aspects of an SLA, and do not consider those technical aspects within the context of a complete SLA. Additionally, most service negotiation frameworks are service specific and the possibility of generalised service negotiation is not considered.

Many service negotiation frameworks, for example [100] and [154] focus on the internal decision processes of the participants, with only a rough specification of participant interactions. Often, a custom Remote Procedure Call (RPC) mechanism is used for interaction as in [155], and usually this is specific to the delivery of one particular service type. Typically, services are distributed by a single mechanism as in [131],[156] or [157]. An open, standard, extensible and flexible communication model should be adopted to support interaction for different service types between heterogeneous service management platforms.

Service negotiation frameworks that have been developed for use with telecommunication based services are typically specific to connectivity services, such as in [154],[158] and [84], assuming upper layer "application" services are managed using a separate system. This approach presents problems if the network is to be actively controlled for application service delivery. An SLA negotiation framework should be flexible enough to handle any type of that is service delivered over, or provides a telecommunication network.

If the scope of services is limited to connectivity alone, another problem is found in that many service negotiation architectures rely on assumptions of a wired network, with static connectivity, and a limited range of providers [60][159]. When wireless networks are addressed, such as in [160], the architectures often assume a homogeneous technological environment or a single administrative body. At best, a pre-existing roaming agreement is assumed between domains [161][162]. Architectures operating under such assumptions are generally not appropriate for the mobile, ubiquitous case where connectivity is highly dynamic and the ability to provide services is not restricted to large operators. In a ubiquitous environment, a service provider may have no information about a roaming user, requiring stronger supporting security and business relationship processes. The Internode project [160] attempts to support nomadic users, albeit for a single service type and pre-existing connectivity agreements with providers.

The web services framework originated from a client-server model with synchronous request-response interactions[163]. Such an operation model does not support rich multi-party interactions and complicated scenarios required to support a range of service distribution models.

A service delivery platform named the IP Multimedia Subsystem (IMS) [164] is currently being developed by fixed and mobile telecommunication providers. Faced with an increasing number of services delivered above a commoditised network, IMS was designed for network operators in an attempt to keep multimedia services in the control of network operators. An advantage here is the integrated network support for QoS, however the controlled approach of IMS does not benefit from open service negotiation, provider flexibility or competitive freedom, as the network operator is in strict control of all the services delivered by the system.

As can be seen, there has been solid research in the domain of agent-based service negotiation for telecommunications. However, automated and ubiquitous delivery of generalised services has not been addressed using an open market approach. The aim of this work is to provide appropriate tools and techniques to manage such service delivery for new generation telecommunication-based services.

2.9 Summary

The initial section of this chapter introduce the reader to a range if telecommunication-based services, discuss the ways in which these services operate, and the way different services fit together to deliver a complete solution. Because all services discussed in this dissertation either rely on, or provide network connectivity, the group of network connectivity services is further analysed. The key QoS characteristics are discussed, along with techniques for delivering QoS-based connectivity.

Continuing the discussion of service delivery, techniques for resource allocation, service pricing and service charging are addressed, as these functions may influence the way agreements for services are structured and/or distributed. In line with the motivation of this work, the concept of ubiquity in service delivery is then analysed. Finally, the current art in describing SLAs is discussed, and an analysis of SLA description and negotiation frameworks is provided.

In the following chapter, we address the requirements in managing future generation services and introduce the proposed solution.

INTENTIONALLY BLANK

Chapter 3

Managing Future Generation Telecommunications Services

Through the study of current work in telecommunications, and in collaboration with industry, this work has one key motivation:

To enable automated management of future generation telecommunication-based services

Chapter 1 discusses the functional areas that should be addressed to sustain an environment in which automated service management may occur. This work, and the remainder of this dissertation addresses two specific areas within the overall motivation:

1. A flexible service level agreement description language
2. A secure, open, flexible SLA negotiation framework.

This chapter outlines the requirements of the above two areas in the context of existing work, and a proposal to address such requirements.

3.1 Solution Requirements

The following set of requirements ensure an SLA specification language and negotiation framework is suitable for current and future telecommunication services. The requirements, and the reasons for their inclusion are described below.

3.1.1 SLA Description Language

A Service Level Agreement description language must have the following characteristics:

- Semantic – There is a clear need for semantics in SLA descriptions[138]. Components of an SLA should be described with semantic relationships that do not require natural language understanding and are not open to misinterpretation. The purpose of formal semantics is as follows:
  - ✧ To establish a universal understanding of concepts used on an agreement
  - ✧ To support reasoning over concepts to enable the discovery of inferred relationships
  - ✧ To support validity checking to ensure any SLA is consistent with all knowledge

about the concepts used on the SLA. If a semantic description is not part of the agreement, the possible "correct" states of each type of agreement has to be derived through some other means.

- Machine-intelligible – A goal of this work is to facilitate automation. An SLA shall be able to be parsed and understood by a computer and the computer should be able to reason over its contents. If an SLA can not be easily interpreted by a computer, a user must be involved and the benefits of automation are lost.
- Open Standards – To promote interoperability and to reduce duplication of effort, the language used to describe the service level agreements and their information model should be a well maintained and managed open standard. This is to facilitate easy adoption and extension of the information model as appropriate.
- Flexible and Extensible – The current range of network related services that are available is large and rapidly expanding as the "information age"[1] matures. A method of describing SLAs should be flexible and able to describe agreements for any type of current and future service. This can be achieved by ensuring:
  - ✧ High de-coupling – Concepts should be grouped into related sets with minimal interdependence. One set of concepts should be *able* to use another to achieve a goal, but should remain independent and not *enforce* use of those concepts. This allows the use of alternatives when appropriate to increase the re-usability of the concept set.
  - ✧ Distributed adoption of service descriptions, not standardisation – Service delivery in an open market should be agile. The acceptance of a service description as a standard by some standards body achieves high interoperability, however the delays incurred are considered unacceptable for this work. If two parties agree on the formal semantics defining a service description, they should be able to use that description in agreements and share that specification easily with others, without waiting for standards approval.
  - ✧ Separation of technical service specific data and non-functional contract specific information on a service agreement. This enables a technical service specification to be reused in different business relationships.
- Minimalist – To remain generic, an SLA description language should specify the minimum number of concepts to provide sufficient structure, while being extensible and allowing the addition of concepts as required for final domain specific SLAs.
- Independent – To remain applicable through advances in technology, the SLA description language should not be tied to any mechanism which uses the Service Level Agreements, such as service provision, service delivery or negotiation mechanisms.

Take the example of grid services. There are many projects working on the capability to deliver grid services[7,8], however different mechanisms for scheduling and signalling are used, and they are often tied directly to the way a service is described. Such coupling creates incompatibilities between platforms. To facilitate interoperability, the components that make up the delivered grid service should be well specified through an abstract interface to the service – That is: A well defined service specification used in an SLA.

- Standalone – A final SLA should be comprehensive. Delivery of a service specified in an SLA should not require additional information that is not available from the SLA. An SLA should specify at least: the entities involved, a description of the service provided, a schedule for the service, commitments by the provider and consumer, and violation outcome for the commitments.
- Secure – Service Level Agreements should be non-repudiable. This requires a method to prove that all involved parties have understood the SLA, and that they agree with and accept the obligations on the SLA. Additionally, the semantics of an agreement should be immutable once the agreement is ratified. For example, if an agreement uses the term "price", it should not be possible to change the semantic meaning of "price" after an agreement has been signed. An SLA specification language should allow semantics to be "locked" for a given SLA, whilst also allowing future change to keep in step with developments in background legislation and business models.
- Grounded – A service level agreement typically specifies limits on real-world parameters such as performance measurements. An SLA specification language should facilitate the comparison of agreed targets to measurable outcomes.
- Compact – Service Level Agreements used in the QDINE project are to be negotiated across telecommunication networks. A compact description of the SLAs is required so as to not place excessive traffic on the networks supporting the negotiation process.

3.1.2 Negotiation Framework

Below are the requirements of an appropriate negotiation framework that enables distribution of the above SLAs, using any negotiation mechanism, supporting ubiquitous mobility for roaming users:

- Autonomous Responsibility – The framework should enable each party to maintain autonomous control over their own resources. For example, a user should be able to receive any available service, so long as the charges can be paid. It is typically the case for current telecommunication services, that users are constrained or managed

7 http://www.unicore.org/

8 http://www.globus.org/toolkit/

by a set of usage policies. Often because there is some manual process required to use a given service. As service configuration becomes automated, users should not be restricted by manually updated usage policies, but should have access to any available resources for which they can pay.

A further consideration of autonomy is that the participants in the framework communicate in a peer-to-peer fashion where no party is entirely reliant on another for participation.

- Security – A Service Level Agreement has value to the stakeholders of a service. A solid, secure communication mechanism is required to ensure the value represented by SLAs cannot be disputed. Additionally, the negotiation framework may transmit sensitive information that should be kept confidential. The requirements for security are as follows:
  - ✧ Ensure positive identification of players involved.
  - ✧ Enable the transmission of sensitive data, ensuring the secrecy of such information is respected as required.
  - ✧ Ensure the integrity of information during transmission.
  - ✧ Enable provable checking of message authenticity and validity when required.
  - ✧ Ensure messages are non-repudiable.
  - ✧ Provide complete and well defined processes, considering potential attacks on the system.
- Separation – A negotiated Service Level Agreement should be separate from the negotiation mechanism. That is, the framework should be able to support the negotiation of a full range of SLAs, independent of the type of service the SLA describes.
- Flexibility – All possible services do not fit into one delivery model. Depending on the supply and demand profile for the service, the configurability of the service, or the business model behind the service delivery, different negotiation mechanisms may be required. The service negotiation platform should support the use of various market mechanisms as applicable to a provided service.

  For service pricing, charging and billing, it is not appropriate to specify one single charging mechanism, pricing algorithm or billing system for all possible telecommunication based services. What is required is a generic framework that can support any charging mechanism, is independent of the pricing mechanism and can interact with any billing system.

  A further aspect of flexibility is related to patterns of interaction possible by participants. To allow flexibility, an agent should not be restricted to synchronous request/response interactions for example, but should be able to send single asynchronous messages, or messages as part of a more complex, and potentially ongoing interaction. The importance of flexibility in managing service interactions is emphasised by Paurobally and Jennings in [163].

- Freedom – The Service Negotiation framework should be open for participation by any entity. This is to encourage the development of novel business processes. To participate in a framework with other agents, some external constraints are typically placed on an agent, such as the use of a common language or the support of a core set of interactions. These external constraints should only apply the minimum restriction to achieve the desired behaviour. Additionally, the framework should not enforce or require any particular internal behaviour of a participating agent. An agent should be free to adopt any decision making model desired, and behave how it sees fit. An agent should conform to any external constraints, but may choose to violate such constraints and be penalised accordingly.
- Supervision – In some situations, strict control over an agent's external behaviour may be desired to ensure external constraints cannot be broken. A facility should exist that provides strict supervision of agents if required. For example, the allocation of a service using an exchange market mechanism may require all participating agents to strictly follow the market protocols. To enforce such a requirement, a supervisor may be employed to proxy all interaction and only admit conforming messages.

  In line with the requirement of freedom, such supervision should not be mandatory, but should be available within the framework if required.
- Open Standards – Communication between entities should be performed via an open, standard mechanism. This is to aid interoperability and ease adoption.
- Ubiquitous – The framework should facilitate ubiquitous service provision across administrative and technologically heterogeneous networks. Ubiquity is discussed in detail in section 2.6. Of concern here is the timely management of ad-hoc business relationships, supplying involved service providers with exactly the information they need to provide the service and be sure that obligations are met. This work is not concerned with technical handover required for the transfer of a service.
- Automated – The framework should introduce minimal continuous administrative overhead for participants of the framework. The motivation for this work is *automated* service delivery. If large administrative effort is incurred to create this automation, many benefits of automation are lost.

3.2 The QDINE proposal

Identified by the above requirements and the existing work explored in chapter 2, there are two problems this work aims to address:

1. There is currently no service level agreement description technique that:

- can be used to describe *any* telecommunication related service, and
- can be used independently of a particular SLA management system, and
- allows automated reasoning over created agreements using common tools and techniques, and
- provides a non-repudiable contract, addressing security issues of the SLAs.

2. There is no current service negotiation framework that:

- supports automated negotiation of service level agreements
- for any type of service
- using any negotiation mechanism appropriate, and
- gives participants autonomous responsibility over their resources, and
- provides a secure environment for interaction, supporting strict supervision if desired.

It is proposed that a system of intelligent agents using knowledge-based communication, a standard set of core interactions, and negotiation mechanisms applicable to the context of the delivered service can be employed to automatically establish binding service level agreements for generalised services. To facilitate automated reasoning, Service Level Agreements are described using a formal knowledge representation language.

Correspondingly, this work makes two main contributions that are described in the following two chapters:

1. A set of formal ontologies that may be used to semantically describe secure service level agreements for any application domain.
2. A multi-agent system providing an open market where services can be discovered, participants identified, and negotiation performed using context specific mechanisms.

These two contributions are developed in accordance with the requirements described above.

Chapter 4
The SLA Ontology

4.1 Introduction

Chapter 1 introduces the main functional areas of an automated and ubiquitous service delivery platform. This chapter addresses one such area; that is, the need for a flexible yet formal method for describing service agreements between parties, outlining the terms of a service provision. The requirements of an appropriate Service Level Agreement description language are discussed in chapter 3.

Presented are a set of ontologies, a concrete formalism and a suggested use which together allow a description of Service Level Agreements while conforming to the requirements outlined in chapter 3. Much of the content of this chapter was first published in [165]. The SLA ontology follows the IETF approach to SLAs where an SLA includes non-functional information, as well as a service specification describing the functional information of the service.

Following the goals of the ontology described below, section 4.2 outlines the need for semantic service level agreements. Section 4.3 follows with a description of the techniques used to express the Service Level Agreements in a concrete form. Each of the created ontologies are outlined in section 4.4, followed by a typical use of the ontologies for creating Service Level Agreements.

4.1.1 Goals

The goals of the SLA Ontology are to:

- Allow entities to create Service Level Agreements with asserted semantics.
- Support reasoning about temporal concepts on SLAs.
- Allow the inclusion of both non-functional (eg. Pricing) and functional (eg Throughput) components on an SLA.
- Provide a formalism to describe comparisons of the actual delivered characteristics of an enacted service to the targets specified on an agreement.
- Maintain a separation between the ontology definitions described and any particular use of the ontologies.

- Illustrate a suggested way in which the ontologies may be used.

4.1.2 Out of Scope

The SLA Ontology does not:

- Describe a definitive list of service parameters applicable to any particular SLA domain.
- Provide a complete prescriptive methodology for using the ontology to fulfil any purpose.

4.2 Semantic Service Level Agreements

To form an agreement between two or more parties, all parties should have a common understanding or "shared semantics" of the terms expressed in the agreement.

Determining a shared semantics for a particular term, requires that the term first be unambiguously identified, then the semantics of that term can either be assumed as common knowledge, or asserted through a formal definition of its place within a formally described domain of knowledge.

Within human understanding, natural language is typically used to identify concepts, and the semantics are assumed to be common knowledge; as such, misunderstandings are common and are an accepted component of "human error". To correct misunderstandings, humans perform a disambiguation process through the exchange of information between parties, often using less ambiguous terms from a natural language. Concept identification is typically considered within a particular universe of discourse or scope. The scope of human concept identification is typically defined by the use of a word within a language and in a particular context.

Artificial intelligence has not matured to a stage where computers can reliably perform concept disambiguation or semantic understanding based on natural language. Some formal method is required to uniquely identify concepts within a particular scope and to describe the knowledge about those concepts.

Ontologies are used to describe a conceptualisation of knowledge from a particular domain. This knowledge is typically represented as "things" and the "relationships between things". Additionally, it is often desirable to describe rules governing the relations between things. Typical ontological concepts include Individual, Class, Relation and Rule. Relationships may be between classes, between classes and individuals or between individuals.

Methods to describe machine comprehensible ontologies, along with tools for reasoning over these ontologies are available. Additionally, concepts within the scope of the ontologies can be uniquely identified. Consequently, an ontological approach to describing SLA semantics is used here.

4.3 A Concrete Formalism

The ontology specification language used within this work is OWL-DL. OWL-DL is an OWL sub-language that is based on Description Logic[166]. OWL-DL allows definition of ontologies with guaranteed decidability: that is, computational queries are tractable in finite time. OWL-DL is standardised by the W3C and is based on Semantic Web Technologies, thereby maximising interoperability. It has multiple open source inference engines and development tools available.

The tool used for ontology development within this work is Protégé[167]. Protégé is a well maintained, open source, frame-based ontology editor with a plug-in for OWL in addition to various visualisation and other supporting tools.

4.3.1 Restricting OWL Flexibility

To achieve a level of flexibility and interoperability desired for Internet ontologies, OWL operates under an open world assumption, allowing inheritance and extension of described concepts. An open world assumption means no assumption can be made on anything that cannot be inferred. "If I only say that object X is an apple, it does not mean it is not an orange". Such freedom can potentially introduce significant problems when attempting to define a secure and binding agreement between two parties. That is; a concept used on an OWL-based SLA may later have additional information attributed: altering the semantics of the concept and potentially the validity of the agreement.

The following techniques may be used to restrict the flexibility of concepts described in OWL when required to ensure valid and binding agreements:

- The use of OWL concepts can be restricted to only those from namespaces that have been accepted and adopted by all parties involved. This is discussed further in section 4.5.1.
- The ways in which concepts from adopted namespaces can be used may also be restricted. This involves many techniques. Some of which are outlined below:
  - ✧ Define rules over concepts through a constraint language, such as SWRL.
  - ✧ Where required, restrict class membership for individuals. Achieving this has multiple possibilities:

- Restrict the range of properties within the property definition.
- Restrict the range of properties when used within certain classes. This is done in the class definition that uses the property. ($\forall$ and $\exists$ restrictions)
- Assert disjointedness between classes where appropriate.
- If a restriction is required to constrain an Individual to be a member of *only* some specified class, the techniques listed above are not sufficient, as they do not limit an Individual from being assigned to multiple classes. If such a restriction is required, SWRL rules can be used. The following rule may be applied to a specific individual, or every individual from a class. The rule restricts every `rdf:type` defined for an individual to a particular OWL class. The following construct may be used:

  $$MyThing(?th) \wedge rdf{:}type(?th,?t) \Rightarrow swrl{:}sameAs(?t,MyThing)$$

  Given $MyThing(?th)$ is an SWRL class atom, stating object th is of the class `MyThing`; $rdf{:}type(?th,?t)$ is a property atom, stating the value of the RDF property `type` for object th is t; and $swrl{:}sameAs(?t,Mything)$ states t must be the same individual as $MyThing$; that is, the class `MyThing`.

  Such a rule can be expressed in the RDF syntax of SWRL, and not in the OWL syntax because in the OWL syntax for SWRL, the `sameAs` atom and `individualPropertyAtom` both require concepts from the OWL language. `rdf:type` is an RDF concept, not an OWL concept.

4.3.2 Semantic Web Dynamics

A key feature of semantic web based ontologies is that they enable a distributed, re-usable description of knowledge. This feature brings with it a potential implementation problem: The specification of an externally referenced OWL resource may change over time. That is, the publisher of an ontology may change any of the definitions contained within the published ontology at will. If agreement to an SLA has been based on a published specification at a particular instance in time, and the specification changes, what is the impact on the validity of the negotiated Service Level Agreement? The SLA should be in the context of the ontology version in which it was signed, as well as in the context of the norms of the market in which it was signed and in the context of the laws that prevailed at the time. Market Norms are discussed further in chapter 5.

The problem of dynamic definitions is compounded by the fact that semantic referencing may occur over numerous "semantic hops". For example, suppose that an SLA uses the term "salePrice", available on an imported ontology, and the "salePrice" from the imported ontology uses the term "cost" from one of its imported ontologies.

What happens to the validity and repudiability of the original SLA if the publisher of the ontology that defined the "cost" concept changes the definition of cost??

The following technique may be adopted to address this issue. This is the solution used by the QDINE framework, discussed in chapter 5.

1. Previous to signing an SLA (SLA binding in the QDINE framework §5.4.4.1), a secure hash is generated of all ontology source files which are used either directly on an SLA, or indirectly through some semantic link from the directly used ontologies. All ontologies referenced at any link depth must have a hash built. The structure of these hashes is described in §4.4.2.
2. The ontology sources are saved locally and linked to the created hash values. The ontology sources need only be saved once while a hash remains valid, and the hash value can be reused on multiple SLAs. When the ontology source changes, a new hash is created and linked to a saved copy of the new ontology source. The old source and hash value is retained for existing SLAs.
3. Service Level Agreements are constructed using the URIs of the original ontology source components. All SLAs are a subclass of `NonRepudiableThing` (§4.4.2.1), and hashes of all ontologies used are attached to the SLA via the `has-OntologyHash` property defined on `NonRepudiableThing`.
4. If both parties sign an SLA containing an ontology hash, they implicitly state their acceptance of the included hash values of the used ontology sources.

Adopting the above technique, the ontology source from the time of agreement creation can be obtained for any created SLA, and the acceptance of this version of the source can be verified. This feature provides two main features. First, the semantics of the SLA concepts can be traced to a particular instance in time. Second, if the value for some property has been specified on an ontology source, and that value is used by an operational SLA, then the value at the time of SLA agreement can be obtained for use while the SLA is in force. These original values will remain available even if a referenced ontology source changes

The above technique is not limited to use by SLAs. As discussed in section 4.4.2, the ontology hash technique is applicable to any OWL individual that is required to be non-repudiable.

4.4 Created Ontologies

The following ontologies were built according to the ontology engineering process described in [168]. To achieve flexibility, the service level agreements have modular components, beginning with generic base components, but allowing the addition and extension of concepts towards a domain specific purpose. Each ontology definition file is designed to be decoupled from other ontologies, meaning only very related classes and properties are kept in the same ontology definition file. The source from all the created ontology definition files is included in appendix A.

A set of generic ontologies is provided as a framework for building service level agreements. They are an open set of specifications, each defining specific concepts in the domain of Service Level Agreements. However, they are designed to be generic and may be used satisfactorily in any other knowledge domain.

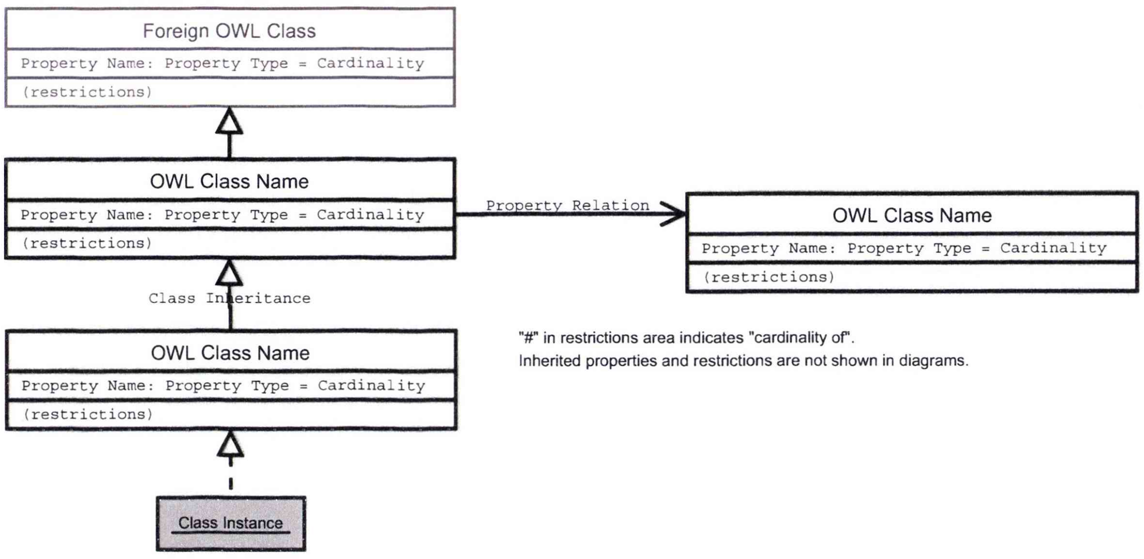
Each of the generic ontologies can be seen as a knowledge building block; for use as a component in a larger specification. They outline what *can* be done but are not meant to be an exhaustive list of possibilities.

4.4.1 Ontology Diagrams

Diagrams are used throughout this chapter to illustrate the ontologies created. The diagrams are similar in semantics and appearance to UML Class diagrams, with the following differences:

- The cardinality of an OWL property is listed in the space where the default value of an attribute is described in an equivalent UML class diagram.
- Restrictions defined on OWL classes are listed in the area where methods are described in a UML class diagram.

Figure 14 describes the layout of the diagrams used throughout this chapter.



Foreign OWL Class
Property Name: Property Type = Cardinality
(restrictions)
OWL Class Name
Property Name: Property Type = Cardinality
(restrictions)
Property Relation
OWL Class Name
Property Name: Property Type = Cardinality
(restrictions)
Class Inheritance
OWL Class Name
Property Name: Property Type = Cardinality
(restrictions)
"#" in restrictions area indicates "cardinality of".
Inherited properties and restrictions are not shown in diagrams.
Class Instance

Figure 14. Ontological Diagram Layout

4.4.2 Ontology Hash Ontology

As discussed previously in section 4.3.2, a published ontology specification document may change over time. Changes to a published document can introduce semantic ambiguity to globally unique URIs, since a concept referred by a URI may have different asserted semantics at different times. OWL includes a "versionInfo" annotation property which can be used to indicate the version of any OWL resource, thereby removing semantic ambiguity. However, OWL does not define a suggested use of the versionInfo property, and use of this property cannot be enforced on the publisher of an ontology.

The Ontology Hash ontology allows the capture of a "snapshot" of one or more ontology specifications, removing the temporal ambiguity on semantics without relying on the document publisher to attach correct version information to every published ontology component.

Figure 15 outlines the classes and relationships in the Ontology Hash ontology. The complete OWL specification for the Ontology Hash ontology is included in appendix A. 1.

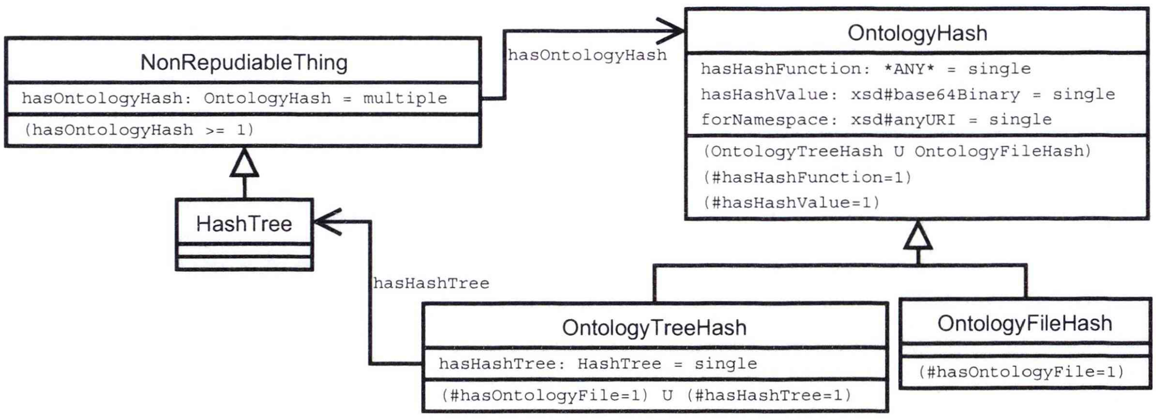

NonRepudiableThing
hasOntologyHash: OntologyHash = multiple
(hasOntologyHash >= 1)
hasOntologyHash
OntologyHash
hasHashFunction: *ANY* = single
hasHashValue: xsd#base64Binary = single
forNamespace: xsd#anyURI = single
(OntologyTreeHash U OntologyFileHash)
(#hasHashFunction=1)
(#hasHashValue=1)
HashTree
hasHashTree
OntologyTreeHash
hasHashTree: HashTree = single
(#hasOntologyFile=1) U (#hasHashTree=1)
OntologyFileHash
(#hasOntologyFile=1)

Figure 15: The Ontology Hash Ontology

4.4.2.1 NonRepudiableThing Class

OWL individuals that require a “snapshot” of their semantics may be assigned to the `NonRepudiableThing` class in addition to any other classes to which they may belong.

All RDF resources that are referenced within a `NonRepudiableThing` are required to have an `OntologyHash` (§4.4.2.2) generated for their defining XML namespace. The OWL, RDF, XML, XSD and RDFS namespaces may be excluded from the list of hashed namespaces, as they are governed by standard bodies and the semantics of the components described are static for a given namespace.

An `OntologyHash` is associated to a `NonRepudiableThing` by the `has-OntologyHash` object property. Each `NonRepudiableThing` must include at least one `OntologyHash` and may include multiple. The range of the `has-OntologyHash` property must be a anonymous OWL individual. That is, it must be unidentified, specified in-line between the opening and closing `hasOntologyHash` tags, and cannot be a foreign referenced resource.

In Example 5, for “testSLA” to be a valid non-repudiable thing, an ontology hash must be created for the namespaces corresponding to the “sla” and “onthash” prefixes, and the “http://example.com/ont/test.owl” namespace.

```
<sla:SLA rdf:ID="testSLA">
  <rdf:type="http://qdine.it.uts.edu.au/ontologies/ontHash.owl#NonRepudiableTh
ing">
  <sla:hasConsumer rdf:resource="http://example.com/ont/test.owl#user1">
  ...
  <onthash:hasOntologyHash>
    <onthash:OntologyTreeHash>
      ...
    </onthash:OntologyTreeHash>
  </onthash:hasOntologyHash>
</sla:SLA>
```

Example 5: Use of the `NonRepudiableThing` *class*

4.4.2.2 Ontology Hash Class

An `OntologyHash` is a secure, unique digest representing an ontology identified by some URI namespace.

All `OntologyHash` individuals must be anonymous OWL individuals and specified inline. This constraint ensures the Ontology Hash can not have additional properties arbitrarily applied from outside the resource definition, as is possible with identified individuals. The `OntologyHash` class has two subclasses: `OntologyFileHash` and `OntologyTreeHash`.

An `OntologyFileHash` defines a secure digest for one single XML namespace. It must include the namespace URI for which the hash value is built, the hash algorithm used and the generated hash value. The suggested source for generating the hash value is the entire binary contents of the file which defines the concepts identified by the hashed namespace. Typically, the source file for a namespace is located at the specified namespace URI and accessible over the Web, however this is not a requirement specified by XML, RDF or OWL.

To generate the hash value for an `OntologyFileHash`, a secure digest of the source data is generated, using the specified hash algorithm such as SHA-1, SHA-256 or WHIRLPOOL [169][170]. This digest is then encoded in base64[171] and stored as the hash value.

An `OntologyTreeHash` defines a secure digest for a tree structure of namespaces. The hash value generated for an `OntologyTreeHash` will change if the source changes for any of the included namespaces. The hashing algorithm and generated hash value are mandatory components of an `OntologyTreeHash`. In addition to these mandatory components, an `OntologyTreeHash` may optionally include the following:

- a reference to one `HashTree` (§4.4.2.3) representing the tree of namespaces used to build the hash value, and/or

- a namespace that specifies the root of a hash tree, built using the standard ontology tree hash layout (§4.4.2.4)

Either a root namespace or hash tree must be specified.

The hash value for an OntologyTreeHash is generated as follows:

1. First, the HashTree for the OntologyTreeHash is obtained as follows:
   - If the forNamespace property is populated, a HashTree conforming to the standard ontology tree hash layout(§4.4.2.4) is assumed, using the namespace specified in the forNamespace property as the root of the tree.
   - Alternatively, if the forNamespace property is not populated, the hasHash-Tree property is followed to the referenced HashTree individual.
2. A secure digest of the HashTree individual is generated using the inclusive text string starting at the first "<" of the opening <HashTree> tag and ending at the last ">" of the closing </HashTree> tag.
3. As with OntologyFileHash, the hash value is stored as a base64 encoding of the secure digest.

4.4.2.3 HashTree Class

Within QDINE, Service Level Agreements are designed to be negotiated and distributed across telecommunication networks. The concept of hash trees is introduced to reduce the amount of hash data necessary on every negotiated SLA. A hash tree encapsulates the hash values of numerous ontology sources into a single hash value.

A HashTree is a subclass of NonRepudiableThing. As such, it includes the hasOntologyHash object property and associated constraints.

As described above in §4.4.2.2, the hash value for an OntologyTreeHash is generated using only the *inclusive* text between the opening "<HashTree>" and closing "</HashTree>" tags of the associated HashTree individual. Correspondingly, an additional constraint is placed on HashTree individuals: All hasOntologyHash properties assigned to a HashTree individual must be included inline within the hash tree's defining tags, as in Example 7 (pg. 77). This constraint exists because if a hasOntologyHash property was allowed to be associated outside the hash tree definition, via the rdf:description tag, the hash value generated for an OntologyTreeHash would not include that externally assigned property, as in

Example 6. Examples 6 and 7 offer a semantically identical OWL description of "tree2". However, the hash value generated from "tree2" in Example 6 does not include the second namespace and is therefore incorrect. The highlighted blocks indicate the text that is used to generate the hash value. The XML tag prefixes are omitted for brevity.

```
<sla:SLA>
  <hasOntologyHash>
    <OntologyTreeHash>
      <hasHashTree>
        <HashTree rdf:ID="tree2>
          <hasOntologyHash>
            <OntologyFileHash>
              <forNamespace>
                http://n1.org
              </forNamespace>
              ....
            </OntologyFileHash>
          </hasOntologyHash>
        </HashTree>
      </hasHashTree>
      <hasHashValue>s7f!
rds</hasHashValue>
      ....
    </OntologyTreeHash>
  </hasOntologyHash>
  ....
</sla:SLA>

<rdf:Description rdf:about="#tree2">
  <hasOntologyHash>
    <OntologyFileHash>
      <forNamespace>
        http://n2.org
      </forNamespace>
      ....
    </OntologyFileHash>
  </hasOntologyHash>
</rdf:Description>
```

Example 6: Namespace n2.org WILL NOT be included in the hash value generated. This is INCORRECT

```
<sla:SLA>
  <hasOntologyHash>
    <OntologyTreeHash>
      <hasHashTree>
        <HashTree rdf:ID="tree2>
          <hasOntologyHash>
            <OntologyFileHash>
              <forNamespace>
                http://n1.org
              </forNamespace>
              ....
            </OntologyFileHash>
          </hasOntologyHash>
          <hasOntologyHash>
            <OntologyFileHash>
              <forNamespace>
                http://n2.org
              </forNamespace>
              ....
            </OntologyFileHash>
          </hasOntologyHash>
        </HashTree>
      </hasHashTree>
      <hasHashValue>7J?
9v4g</hasHashValue>
      ....
    </OntologyTreeHash>
  </hasOntologyHash>
  ....
<sla:SLA>
```

Example 7: Namespace n2.org WILL be included in hash value generated. This is CORRECT

To ensure consistent generation of `OntologyTreeHash` hash values, the prefix of the Ontology Hash namespace[9] used within `HashTree` individuals should always be the string "onthash". The use of different prefixes will generate different hash values because the prefixes are included in the data passed to the hashing algorithm.

Hash trees are designed to capture infrequent changes to an ontology definition. The suggested use of hash trees is for the publisher of an ontology to also publish an OWL file including all the hash trees for the ontologies which it has published.

9 http://qdine.it.uts.edu.au/ontologies/OntHash.owl#

4.4.2.4 Standard Ontology Tree Hash Layout

The Standard Ontology Tree Hash Layout is a method of generating a hash value for an ontology namespace, recursively encompassing all the referenced ontologies used within that ontology. `OntologyTreeHash` individuals are not *required* to follow this layout, however it is a recommended strategy.

To indicate that a hash value included on an `OntologyTreeHash` has been generated using the Standard Ontology Tree Hash Layout, the `forNamespace` property on the `OntologyTreeHash` individual is populated with an ontology URI. This URI is the root of the `OntologyTreeHash`. The `hasHashTree` property of such an `OntologyTreeHash` may be left unpopulated, as the `HashTree` structure is assumed to conform to the standard layout constraints(§4.4.2.4.1). If desired, a `Hash-Tree` may be included on a "standard" `OntologyTreeHash` to aid investigation if the hash value does not match when another agent attempts to validate the Ontology Hash. A sample `OntologyTreeHash` conforming to the Standard Ontology Tree Hash Layout can be found in appendix section .

Enforcing the use of the Standard Ontology Tree Hash Layout for every `Ontology-TreeHash` created was considered by this work. This would allow the `hasHashTree` property to be entirely removed from `OntologyTreeHash` class. However, this approach is overly restrictive for adopters of the ontology hash ontology, placing large assumptions on the semantics of the `OntologyTreeHash` content. Additionally, the exclusion of a descriptive `HashTree` provides participants of an interaction with no information on which namespace is in conflict if a hash value mismatch occurs while validating the hash value on an `OntologyTreeHash`. For these reasons, the `has-HashTree` property is kept, but is optional.

4.4.2.4.1 Layout Constraints

The following statements define the constraints that apply to an `OntologyTreeHash` built using the standard ontology tree hash layout.

- The namespace prefix used for the Ontology Hash namespace is "onthash".
- All non functional whitespace is removed. I.e. Any whitespace which doesn't change the meaning of the `HashTree` is removed. For example, there is no whitespace used before or after either an opening or closing tag. All functionally required whitespace must be one single space character " ".
- All named resources used within object properties are referenced using an

rdf:resource location statement using the full URI (eg. rdf:resource="full.namespace#ID") listed within the object property opening tag. One space character " " is used to separate the object property tag name and the rdf resource location statement.

- The first `OntologyHash` listed within the `HashTree` must be an `OntologyFileHash` for the root `forNamespace` URI listed on the parent `OntologyTreeHash`. An `OntologyHash` must be included within the `HashTree` for every RDF resource used from a foreign namespace. These ontology hashes are included in the order that the resources are used within the ontology source file. A namespace can only be included once within the `HashTree`.
- All foreign namespaces used, whose source file described by the namespace refer to no foreign resources must be hashed as an `OntologyFileHash`. All foreign namespaces used, whose source file described by the namespace refer to foreign resources must be hashed as an `OntologyTreeHash`, conforming to the Standard Ontology Tree Hash Layout.
- Within each `OntologyHash` individual, the `forNamespace` property must be listed first followed by the `hasHashFunction` property and then the `hasHashValue` property. If a standard `OntologyTreeHash` explicitly includes a `HashTree`, then every `OntologyTreeHash` individual described within that `HashTree` must also explicitly specify the `HashTree` used to generate the hash value. The `hasHashTree` property must proceed the `hasHashValue` property.

4.4.3 SLA Ontology

The SLA ontology is a mid-level ontology for describing service level agreements. It is a generic ontology, specifying a minimal core set of concepts for describing an SLA. It is designed to be extended into domain specific Service Level Agreements, adding subclasses and extra properties if required, and restricting the implemented properties into specific classes. The SLA ontology imports no other ontologies in order to maintain highly decoupled from unrelated concepts. The complete OWL specification for the SLA ontology is included in appendix A.1.

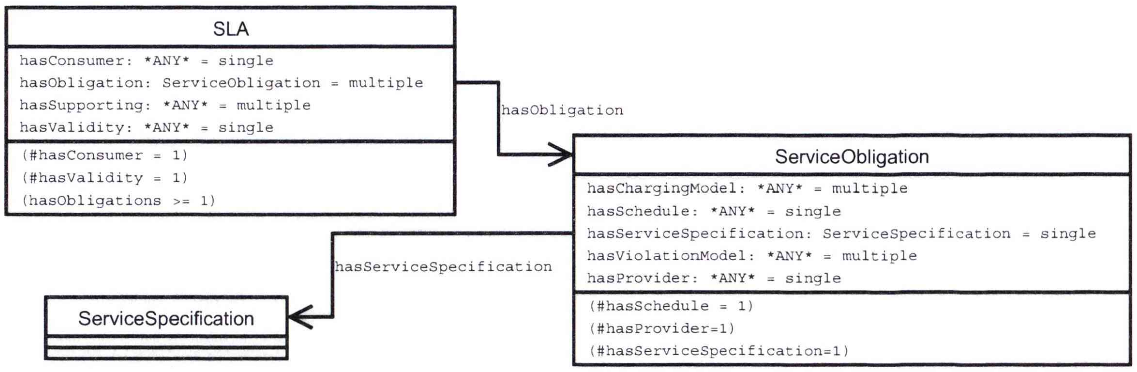

SLA
hasConsumer: *ANY* = single
hasObligation: ServiceObligation = multiple
hasSupporting: *ANY* = multiple
hasValidity: *ANY* = single
(#hasConsumer = 1)
(#hasValidity = 1)
(hasObligations >= 1)
hasObligation
ServiceObligation
hasChargingModel: *ANY* = multiple
hasSchedule: *ANY* = single
hasServiceSpecification: ServiceSpecification = single
hasViolationModel: *ANY* = multiple
hasProvider: *ANY* = single
(#hasSchedule = 1)
(#hasProvider=1)
(#hasServiceSpecification=1)
hasServiceSpecification
ServiceSpecification

Figure 16: The SLA Ontology

4.4.3.1 SLA Class

The SLA class is the parent container used to describe a Service Level Agreement. It specifies the consumer of the SLA, any supporting entities to the SLA (not the provider), a period of validity for the entire SLA and one or more obligations which describe the bilateral commitments made by a provider and the consumer. An SLA may include multiple service obligations for different providers.

4.4.3.2 ServiceObligation Class

An SLA can have multiple Service Obligations, with each obligation describing a set of commitments made by a provider in relation to a single provided service, and any commitments made by the consumer for receiving the service. The `ServiceObligation` class is generic, including concepts which are typical to most service level agreements. A service obligation includes:

- A provider for the service indicating the obliged entity,
- A schedule for the service, identifying when the service is active,
- A formal Service Specification that fully describes what the service is and how it will be provided.
- A Charging model describing the costs for providing the service and how the costs are incurred. That is, the commitment made by the consumer. The charging model should contain any information relating to the pricing or charging for the service. If pricing or charging details are expected to change during the service lifetime, the mechanism by which this change is made should be specified here. For example, if service pricing is renegotiated during the service lifetime, then the negotiation mechanism should be listed here.
  A charging model may not be a financial charge, it may be any reciprocal commitment for the provision of the service.
- A Violation Model indicating a set of commitments made by the provider the

corresponding consequences for violating those commitments. Functional commitments such as throughput guarantees, as well as non-functional commitments such as availability or reliability guarantees, can be addressed through the use of a violation model.

The range of Charging Models and Violation Models used within QDINE are not restricted. It is expected that different models will be appropriate for different service types. As such only the above guidelines to these models are provided. Sample event based violation and charging models have been developed by QDINE for use in experimentation. These are described in section 6.5.1.

The generality of the `ServiceObligation` class lends itself to further refinement for use in a particular domain. This can be achieved through any of the restriction methods available in OWL, such as subclassing, adding properties to classes or restrictions on existing properties, or applying rules to components via SWRL or some other constraint language.

4.4.3.3 ServiceSpecification Class

An OWL individual may be assigned to the class `ServiceSpecification` to indicate that it formally describes the functional properties of a service, for use within service level agreements. The contents of a service specification is not described within this work, as a service specification is specific to the service being described, and the range of supported services is unrestricted. An example service specification, based on an IP connectivity service is specified in section 6.5.3.1.

4.4.3.4 SLA Termination

A service described on an SLA either terminates according to the schedule described on its containing service obligation or terminates at the completion of the entire SLA validity period. A service schedule or SLA validity period should include a predetermined termination time and/or some other alternative termination model. An alternative termination model should clearly describe how a service or SLA can be terminated by the involved parties before its scheduled completion, and the consequences for each potential early termination scenario.

An SLA may be terminated for any number of reasons, and the appropriate termination models for an SLA are dependant on the context of the provided service. An exhaustive list of termination models is therefore not described here. The termination model should be expressed within an SLA at the time of its creation. Four basic scenarios for early termination are described below. These scenarios may be used to form the basis of an early SLA termination model.

1. Requested by Consumer – A consumer may request the early termination of a service or complete SLA. Such a termination may optionally be accompanied by negotiation for a replacement SLA, effectively becoming an SLA renegotiation.
2. Requested by Provider – A service provider may wish to terminate a service or complete SLA before its scheduled expiry. As with a consumer request, a provider initiated termination request may be accompanied by an offer of negotiation for a replacement SLA and so is also an SLA renegotiation.
3. Initiated by Provider – The service provider may choose to prematurely terminate a service at some time without consultation with the consumer. Before doing so, the provider may choose to inform the consumer that the service is about to be cancelled.
4. Supporting connection lost – The QDINE work is concerned with negotiation of services delivered over telecommunication networks. If the connection supporting a service is lost, then after some period of disconnection, the service may be terminated.

4.4.4 Unit Ontology

Service Level Agreements formalise a pair of commitments on providing some service with a particular level of quality. To evaluate the delivery of the commitments, comparisons are required between the agreed target value of some descriptive property and the actual value delivered. That is, we must be able to compare properties that are used to describe an SLA. The Unit ontology is designed to formalise any thing that is comparable, along with the operations that may be performed on such things. Figures 17 and 19 (pg. 83 & 84) describe the classes and relationships in the Unit Ontology. The complete OWL specification for the Unit ontology is included in appendix A.1.

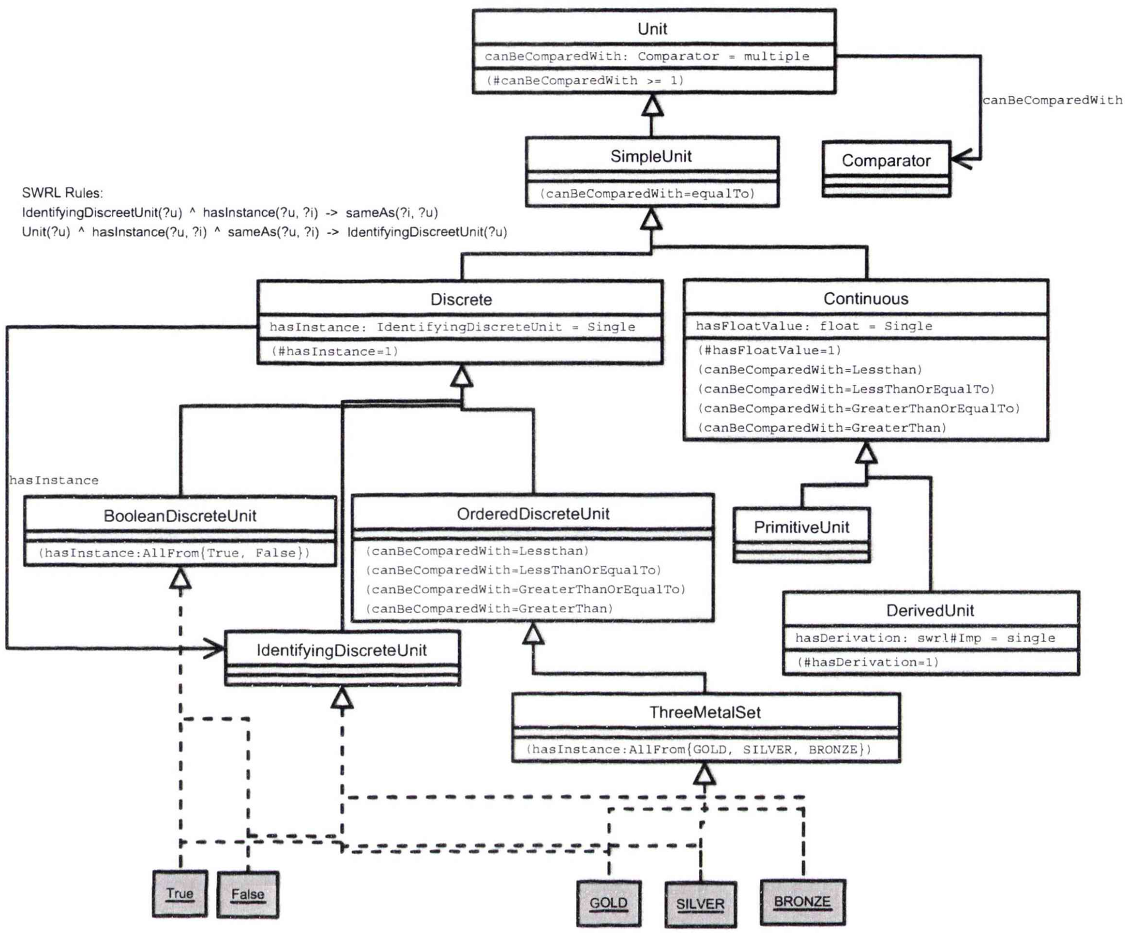

Unit
canBeComparedWith: Comparator = multiple
(#canBeComparedWith >= 1)
canBeComparedWith
SimpleUnit
(canBeComparedWith=equalTo)
Comparator
SWRL Rules:
IdentifyingDiscreetUnit(?u) ^ hasInstance(?u, ?i) -> sameAs(?i, ?u)
Unit(?u) ^ hasInstance(?u, ?i) ^ sameAs(?u, ?i) -> IdentifyingDiscreetUnit(?u)
Discrete
hasInstance: IdentifyingDiscreteUnit = Single
(#hasInstance=1)
Continuous
hasFloatValue: float = Single
(#hasFloatValue=1)
(canBeComparedWith=Lessthan)
(canBeComparedWith=LessThanOrEqualTo)
(canBeComparedWith=GreaterThanOrEqualTo)
(canBeComparedWith=GreaterThan)
hasInstance
BooleanDiscreteUnit
(hasInstance:AllFrom{True, False})
OrderedDiscreteUnit
(canBeComparedWith=Lessthan)
(canBeComparedWith=LessThanOrEqualTo)
(canBeComparedWith=GreaterThanOrEqualTo)
(canBeComparedWith=GreaterThan)
PrimitiveUnit
DerivedUnit
hasDerivation: swrl#Imp = single
(#hasDerivation=1)
IdentifyingDiscreteUnit
ThreeMetalSet
(hasInstance:AllFrom{GOLD, SILVER, BRONZE})
True
False
GOLD
SILVER
BRONZE

Figure 17: The Unit Ontology - Part 1

4.4.4.1 Unit Class

The `Unit` class describes a set of individuals with an explicitly defined set of supported comparison operations. The class name of a unit indicates the unit of measure in the traditional sense – such as Millimetre or Kilogram, and the individuals of the class define actual values of the unit. Figure 18 offers a graphical description of this concept.

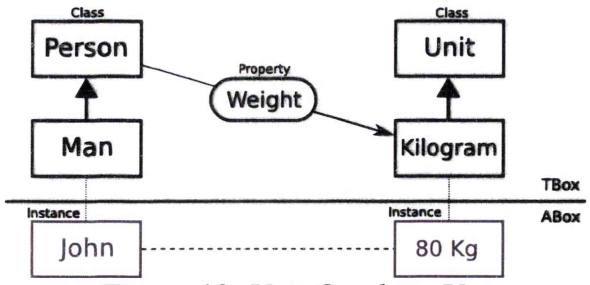
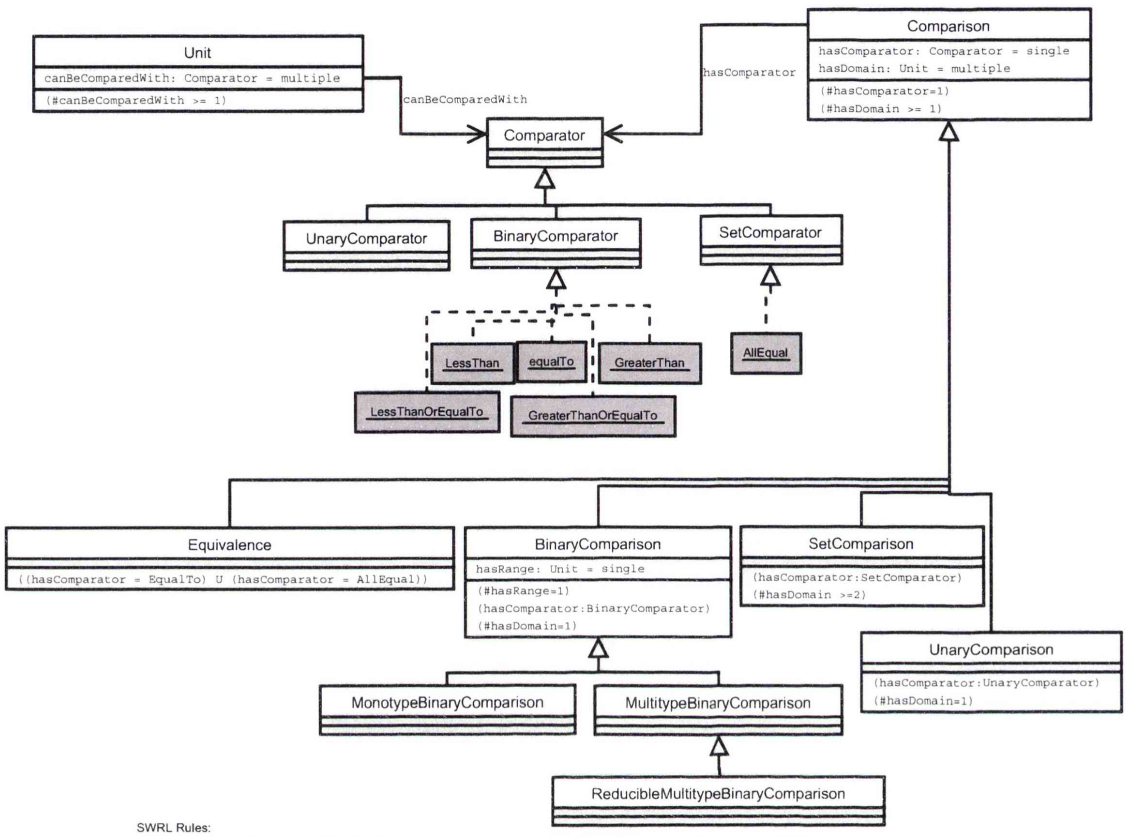


Figure 18: Unit Ontology Use

A `Unit` individual may belong to the `SimpleUnit` class, meaning it has a single property that defines the value for the unit. That defining property can only be bound to immutable objects such as a data literal or `IdentifyingDiscreteUnit` (described below). Simple units are divided into two distinct subclasses; `Continuous` or `Discrete`.



SWRL Rules:
Comparator(?o) ^ Comparison(?c) ^ Unit(?u) ^ hasComparator(?c, ?o) ^ hasRange(?c, ?u) -> canBeComparedWith(?u, ?o)
Comparator(?o) ^ Comparison(?c) ^ Unit(?u) ^ hasComparator(?c, ?o) ^ hasDomain(?c, ?u) -> canBeComparedWith(?u, ?o)

Figure 19: The Unit Ontology - Part 2

The defining property for a `Continuous` unit is a datatype property `hasFloat-Value` that describes the numeric value of that individual. A `Continuous` unit can be either a `PrimitiveUnit` or a `DerivedUnit`. A derived unit is related to one or more other units through a formula described by the `hasDerivation` property. Examples of primitive and derived units are second and minute respectively. Example 8 (pg. 85) illustrates the use of the `Continuous` unit class.

The defining property for a `Discrete` unit is the object property `hasInstance`; the range of which must be an `IdentifyingDiscreteUnit`. The `hasInstance` property of an `IdentifyingDiscreteUnit` is populated with its own identifier. This concept is demonstrated by the unit individual "good" from Example 9 (pg. 86).

Subclasses of `DiscreteUnit` may have a set of identifying individuals created that express the allowed values that an individual from the class may take. For example, a "Service Grade" discrete unit may be able to take the value "Gold", "Silver" and "Bronze". To restrict the allowable values, the "Service Grade" class may have an

"AllValuesFrom" restriction placed on the `hasInstance` slot, listing the three possible individuals allowed as values in that class.

Use of the Unit class is illustrated by the following two examples:

- Example 1 - A specific door ("door1") may be given a description of "width = 90cm". This description may be expressed using the Unit ontology as follows:
  - ✧ An OWL class "Centimetre" is created as a subclass of `ContinuousUnit`.
  - ✧ An OWL class "TwoDimensionalObject" is created, along with an object property "width", with a domain of "TwoDimensionalObject" and a range of "Centimetre".
  - ✧ An OWL class "Door" is created as a subclass of "TwoDimensionalObject".
  - ✧ A "Door" individual is created, given an ID "door1", and has its "width" property assigned to a new anonymous "Centimetre" individual whose `hasFloatValue` property is "90".

```
<owl:Class rdf:ID="Centimetre">
  <rdfs:subClassOf rdf:resource="UnitOntology.owl#ContinuousUnit/>
</owl:Class>
<owl:Class rdf:ID="TwoDimensionalObject"/>
<owl:ObjectProperty rdf:ID="width">
  <rdfs:domain rdf:resource="#TwoDimensionalObject"/>
  <rdfs:range rdf:resource="#Centimetre"/>
</owl:ObjectProperty>
<owl:Class rdf:ID="Door">
  <rdfs:subClassOf rdf:resource="#TwoDimensionalObject">
</owl:Class>
<Door rdf:ID="door1">
  <width>
    <Centimetre>
      <unit:hasFloatValue>90</unit:hasFloatValue>
    </Centimetre>
  </width>
</Door>
```

Example 8: Unit Class example 1 - Door width

- Example 2 - A network service ("myServ") may be given a description of "bandwidth=good". This description may be expressed using the unit ontology as follows:
  - ✧ An OWL class "Quality" is created as a subclass of `OrderedDiscreteUnit`. Five identifying individuals for "Quality" are also created: "Worst", "Bad", "OK", "Good", and "Best"
  - ✧ An OWL class "NetworkService" is created and given an object property "bandwidthQuality" with a range of "Quality".
  - ✧ A "NetworkService" individual named "myServ" is created, and the "bandwidth-Quality" assigned to the Quality individual with the "Good" identifier.

```
<owl:Class rdf:ID="Quality">
  <rdfs:subClassOf rdf:resource="UnitOntology.owl#OrderedDiscreteUnit/>
</owl:Class>
<Quality rdf:ID="Good">
  <rdf:type rdf:resource="UnitOntology.owl#IdentifyingOrderedDiscreteUnit"/>
  <unit:hasInstance rdf:resource="#Good"/>
</Quality>
...
<owl:Class rdf:ID="NetworkService"/>
<owl:ObjectProperty rdf:ID="bandwidthQuality">
  <rdfs:domain rdf:resource="#NetworkService"/>
  <rdfs:range rdf:resource="#Quality"/>
</owl:ObjectProperty>
<NetworkService rdf:ID="myServ">
  <bandwidthQuality rdf:resource="#Good"/>
</NetworkService>
```

Example 9: Unit Class example 2 - Network service bandwidth

4.4.4.2 Comparator Class

Comparators are used to describe comparison operations on Units. Comparators are associated to units through the `canBeComparedWith` property of the `Unit` class. Every unit individual must have at least one `Comparator` defined in its `canBeComparedWith` property.

4.4.4.3 Comparison Class

The `Comparison` class formalises the set of comparison operations possible on `Unit` individuals. A `Comparison` individual describes a comparison operation performed on one or more units using a `Comparator`. Comparisons may result in either a satisfied or unsatisfied (true or false) state.

There are three subclasses of comparison: Set comparisons operating over sets of unit individuals, unary tests on a single unit individual or binary comparisons between exactly two unit individuals.

If both domain and range in a binary comparison are the same class, it is a `MonoTypeBinaryUnitComparison`, if the units involved are of different classes, the comparison is a `MultiTypeBinaryUnitComparison`.

A multi type binary comparison is a `ReducibleMultiTypeBinaryUnitComparison` if and only if the domain and range are both `Continuous` units, at least one is a `DerivedUnit`, and the units are reducible to a common unit through the `hasDerivation` property. An example of a valid `ReducibleMultiTypeBinaryUnitComparison` is a comparison between `Second` and `Minute`.

Every comparison must use a comparator that is appropriate to the objects being compared. This integrity constraint is expressed in Formula 6.

$$\mathit{Comparison}(C) \wedge \mathit{Comparator}(O) \wedge \mathit{Unit}(U) \wedge \mathit{hasComparator}(C,O) \wedge \mathit{hasDomain}(C,U) \Rightarrow \mathit{canBeComparedWith}(U,O)$$

Formula 6: Comparison Validation

4.4.5 Temporal Ontology

The temporal ontology describes concepts related to instantaneous occurrences (The `Event` class) and continuums punctuated by two events (The `Interval` Class). The ontology is created to be complimentary with the Event Calculus proposed by Kowalski and Sergot [172].

In the Event Calculus (EC), fluents[10] are half-open[11] intervals of time, initiated and terminated by events. The state of a fluent at a given time can be determined by predicates and axioms involving events, fluents and discrete points in time. These predicates and axioms are defined in the event calculus.

In the Temporal Ontology, the `Interval` class represents EC fluents and the `Event` class represents EC events. As such, Intervals and Events from the temporal ontology may be used within the Event Calculus for reasoning over temporal concepts expressed on Service Level Agreements. The generic temporal ontology concepts are not bound exclusively to the Event Calculus, they may be used with any compatible temporal logic, such as the interval-based temporal logic [173]. The Temporal Ontology is outlined in Figure 20 (pg. 89). The complete OWL specification for the Ontology Hash ontology is included in appendix A.1.

Two unary comparators: `Occurs` and `IsActive` are defined in the time unit ontology(§4.4.7). These comparators respectively correspond to the "Happens" and "HoldsAt" predicates from EC, implicitly using the current time for the "t" variable in those predicates. Using the `Occurs` and `IsActive` comparators, the real-time occurrence of an event can be determined, and the current state of an interval can be tested.

4.4.5.1 Event Class

An event is a moment in time with no duration. It describes an occurrence at some discrete point in a time continuum. The `Event` class is designed to be generic, and

10 The term "fluent" expresses some object with a state that is dependent on time.

11 A half-open interval (i,j] is inclusive of the final endpoint j, but exclusive of the beginning point i.

subclassed as required by foreign ontologies to fit a desired purpose. For example, an ontology may import the Temporal ontology and create a new subclass of `Event` named "ReceiveEmail" with a property specifying the receiver address.

Many things can be an event, such as the receipt of an email or a specific date and/or time, or the beginning of World War II. A common property of all events is that they may have actions associated which are triggered when the event occurs, and conditions that define the situation under which the event *may* occur. The conditions are not used to indicate the exact situation under which an event is triggered, they are used to help refine an event. That is, some event may exist, but to ensure the event is the exact one we are concerned with, the conditions are applied to test the event conformance.

4.4.5.1.1 DateTimeEvent

The `DateTimeEvent` class is used to represent time moments in the Gregorian calendar as a concrete realisation of the "time" concept from the Event Calculus. Figure 20 (pg. 89) outlines the components of the `DateTimeEvent` class. A `DateTime-Event` occurs every time the largest specified unit of time occurs. For example if a `DateTimeEvent` individual only has the hour and minute properties defined, the event will occur every day on the specified minute and hour. If a larger date/time unit is defined, without defining the smaller units, the smaller units are assumed to be 0 for time based units, or 1 for units of day length and larger. Continuing the above example with only the hour and minute defined, the value of the `hasSecond` property is assumed to be 0.

SWRL rules have been created to ensure valid numeric ranges for properties of a `DateTimeEvent`. For example, if the month specified is Jan, Mar etc, the maximum value for the `hasDays` attribute is 31, if Feb then the maximum is 29. Formulae 7 and 8 illustrate some example rules applied to the `DateTimeEvent` class.

$$\mathbf{DateTimeEvent}(e) \wedge \mathbf{hasMinute}(e,m) \Rightarrow \mathbf{greaterThanOrEqual}(m,0)$$

Formula 7: hasMinute lower bound

$$\mathbf{DateTimeEvent}(e) \wedge \mathbf{hasMinute}(e,m) \Rightarrow \mathbf{lessThan}(m,60)$$

Formula 8: hasMinute upper bound

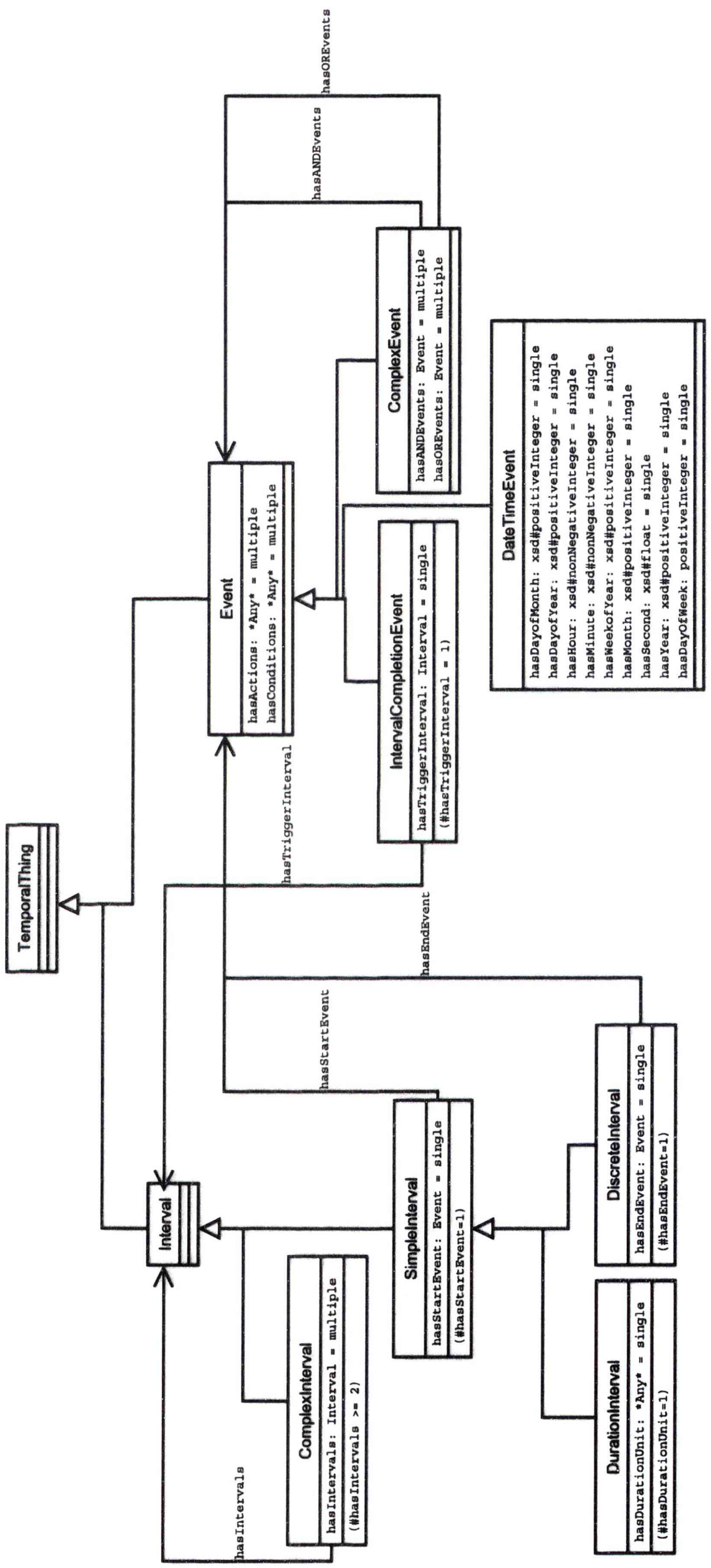

TemporalThing
Event
hasActions: *Any* = multiple
hasConditions: *Any* = multiple
ComplexEvent
hasANDEvents: Event = multiple
hasOREvents: Event = multiple
hasANDEvents
hasOREvents
IntervalCompletionEvent
hasTriggerInterval: Interval = single
(#hasTriggerInterval = 1)
hasTriggerInterval
DateTimeEvent
hasDayofMonth: xsd#positiveInteger = single
hasDayofYear: xsd#positiveInteger = single
hasHour: xsd#nonNegativeInteger = single
hasMinute: xsd#nonNegativeInteger = single
hasWeekofYear: xsd#positiveInteger = single
hasMonth: xsd#positiveInteger = single
hasSecond: xsd#float = single
hasYear: xsd#positiveInteger = single
hasDayOfWeek: positiveInteger = single
Interval
ComplexInterval
hasIntervals: Interval = multiple
(#hasIntervals >= 2)
hasIntervals
SimpleInterval
hasStartEvent: Event = single
(#hasStartEvent=1)
hasStartEvent
DiscreteInterval
hasEndEvent: Event = single
(#hasEndEvent=1)
hasEndEvent
DurationInterval
hasDurationUnit: *Any* = single
(#hasDurationUnit=1)

Figure 20: The Temporal Ontology

4.4.5.2 Interval Class

Similar to an Event Calculus "fluent", the `Interval` class represents a half-open time interval. As such, the end event is included in the interval, but the start event is not. The interval begins immediately *after* the start event. An interval may be a `DiscreteInterval`, explicitly defined by a start and an end event, or a `DurationInterval` whose completion is defined by the passing of some duration after a start event. The run time for a `DurationInterval` is expressed as some measurable unit. As such, the unit expression should include a measurement model. Two example Intervals are:

- The time period between 7:00am and 4:00pm (Discrete Interval).
- The time period between the start of World War II, until 3000 people were killed (Duration Interval).

4.4.6 Metrics Ontology

An SLA consumer will want to be sure a promised level of service is being delivered, requiring the comparison of real world performance data to the promised targets. To describe the use of real world measurements within SLAs, the metrics ontology extends the unit ontology by attaching a measurement model to units. The Metrics ontology also defines a Performance Level that can be used to provide a comparison between a target unit and a real world metric. Figure 21 outlines the metrics ontology, and the complete OWL specification is included in appendix A.1.

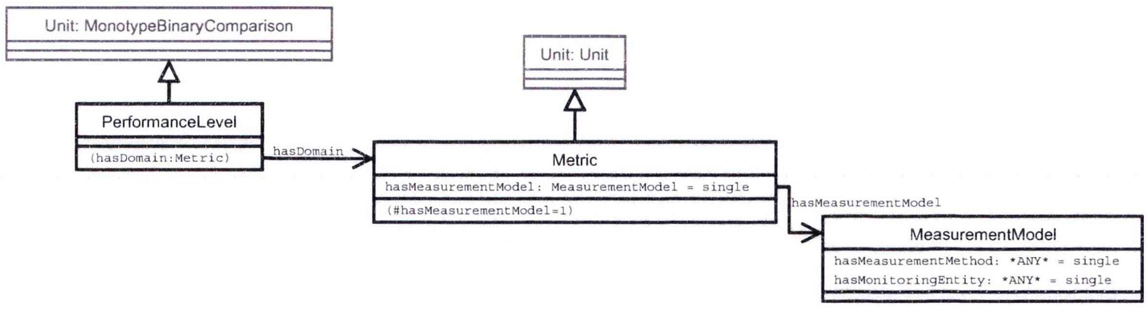

Unit: MonotypeBinaryComparison
Unit: Unit
PerformanceLevel
(hasDomain:Metric)
hasDomain
Metric
hasMeasurementModel: MeasurementModel = single
(#hasMeasurementModel=1)
hasMeasurementModel
MeasurementModel
hasMeasurementMethod: *ANY* = single
hasMonitoringEntity: *ANY* = single

Figure 21: The Metrics Ontology

4.4.6.1 Metric Class

The `Metric` class extends the `Unit` class by defining a method for obtaining a real world measurement of the unit value. As such; an OWL individual assigned to the `Metric` class defines a comparable, measurable thing. A `Metric` specifies a `MeasurementModel` that describes how the real life value of the metric is collected and by whom. This association of a measurement model with a metric is similar to the approach taken in the WSLA work[88].

4.4.6.2 Measurement Model Class

A measurement model defines a measurement method and measuring entity for a metric. The measurement method should precisely describe the measuring technique used to arrive at the value associated to a metric, as well as the schedule of when that value is updated. For example, a metric may be the percentage of dropped, in-profile packets within a 1 minute interval at router queue XYZ. An appropriate update schedule for this metric may be specified as a regular interval in some multiple of 1 minute.

4.4.6.3 Performance Level Class

The `PerformanceLevel` class is a subclass of `Comparison`, enabling the real world value of a `Metric` to be compared with some target unit value. A Performance level is evaluated every time the metric associated with the performance level is updated. Example 10 illustrates the use of a performance level.

```
<PerformanceLevel>
  <hasDomain>
    <BitsPerSecond rdf:ID="SLSThroughput">
      <rdf:type rdf:resource="#Metric"/>
      <hasMeasurementModel> ... </hasMeasurementModel>
    </BitsPerSecond>
  </hasdomain>
  <hasRange>
    <BitsPerSecond><hasFloatValue>3</hasFloatValue></BitsPerSecond>
  </hasRange>
  <hasComparator rdf:resource="Unit.owl#GreaterThan"/>
</PerformanceLevel>
```

Example 10: Performance Level Use

The performance level from Example 10 states that the “SLS Throughput” metric individual, which is of type `BitsPerSecond` must have a real world measured value greater than 3.

4.4.7 Time Unit Ontology

The time unit ontology is an extension to the Unit ontology in the specific domain of temporal units. Most time units are continuous, with the exception of `MonthOfYear`; a discrete unit comprising individuals for the 12 months of the year.

Figure 22 outlines the time unit ontology.

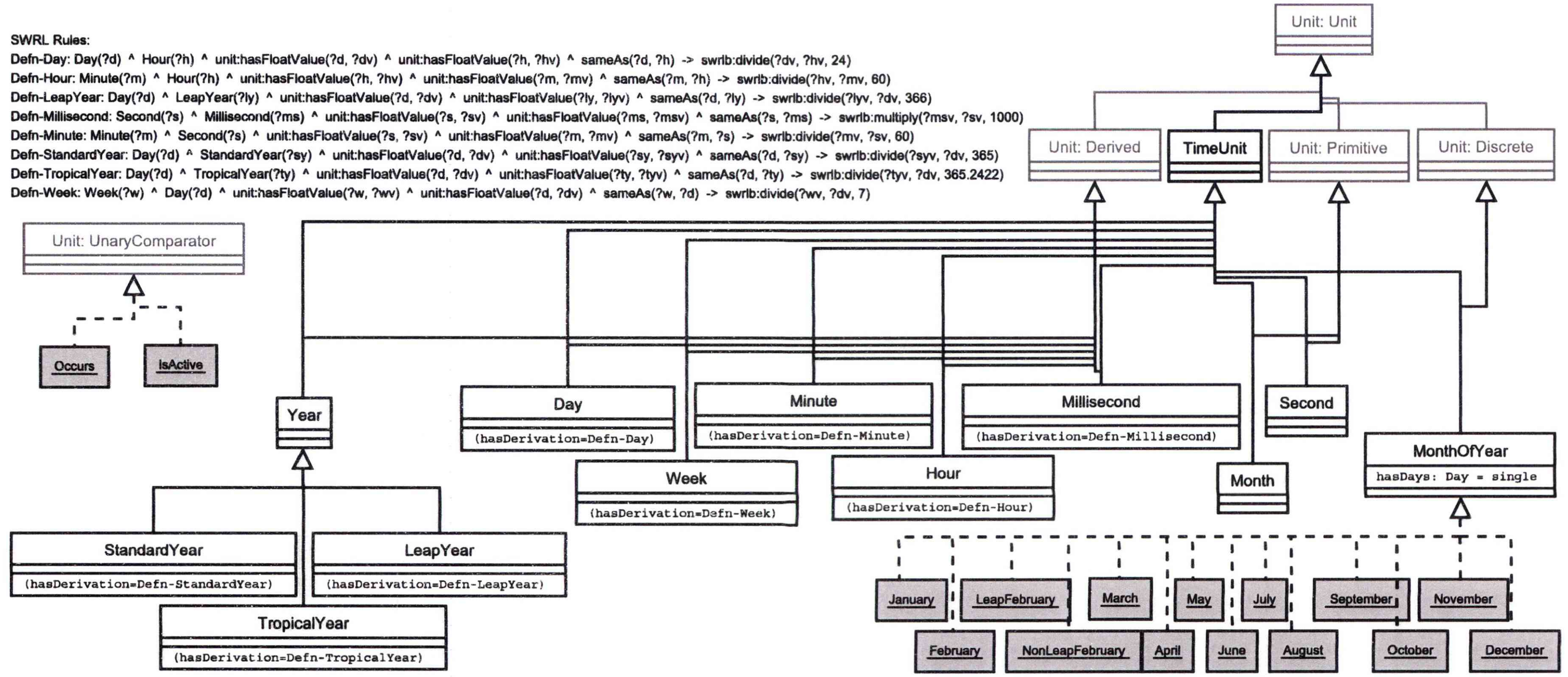

SWRL Rules:
Defn-Day: Day(?d) ^ Hour(?h) ^ unit:hasFloatValue(?d, ?dv) ^ unit:hasFloatValue(?h, ?hv) ^ sameAs(?d, ?h) -> swrlb:divide(?dv, ?hv, 24)
Defn-Hour: Minute(?m) ^ Hour(?h) ^ unit:hasFloatValue(?h, ?hv) ^ unit:hasFloatValue(?m, ?mv) ^ sameAs(?m, ?h) -> swrlb:divide(?hv, ?mv, 60)
Defn-LeapYear: Day(?d) ^ LeapYear(?ly) ^ unit:hasFloatValue(?d, ?dv) ^ unit:hasFloatValue(?ly, ?lyv) ^ sameAs(?d, ?ly) -> swrlb:divide(?lyv, ?dv, 366)
Defn-Millisecond: Second(?s) ^ Millisecond(?ms) ^ unit:hasFloatValue(?s, ?sv) ^ unit:hasFloatValue(?ms, ?msv) ^ sameAs(?s, ?ms) -> swrlb:multiply(?msv, ?sv, 1000)
Defn-Minute: Minute(?m) ^ Second(?s) ^ unit:hasFloatValue(?s, ?sv) ^ unit:hasFloatValue(?m, ?mv) ^ sameAs(?m, ?s) -> swrlb:divide(?mv, ?sv, 60)
Defn-StandardYear: Day(?d) ^ StandardYear(?sy) ^ unit:hasFloatValue(?d, ?dv) ^ unit:hasFloatValue(?sy, ?syv) ^ sameAs(?d, ?sy) -> swrlb:divide(?syv, ?dv, 365)
Defn-TropicalYear: Day(?d) ^ TropicalYear(?ty) ^ unit:hasFloatValue(?d, ?dv) ^ unit:hasFloatValue(?ty, ?tyv) ^ sameAs(?d, ?ty) -> swrlb:divide(?tyv, ?dv, 365.2422)
Defn-Week: Week(?w) ^ Day(?d) ^ unit:hasFloatValue(?w, ?wv) ^ unit:hasFloatValue(?d, ?dv) ^ sameAs(?w, ?d) -> swrlb:divide(?wv, ?dv, 7)

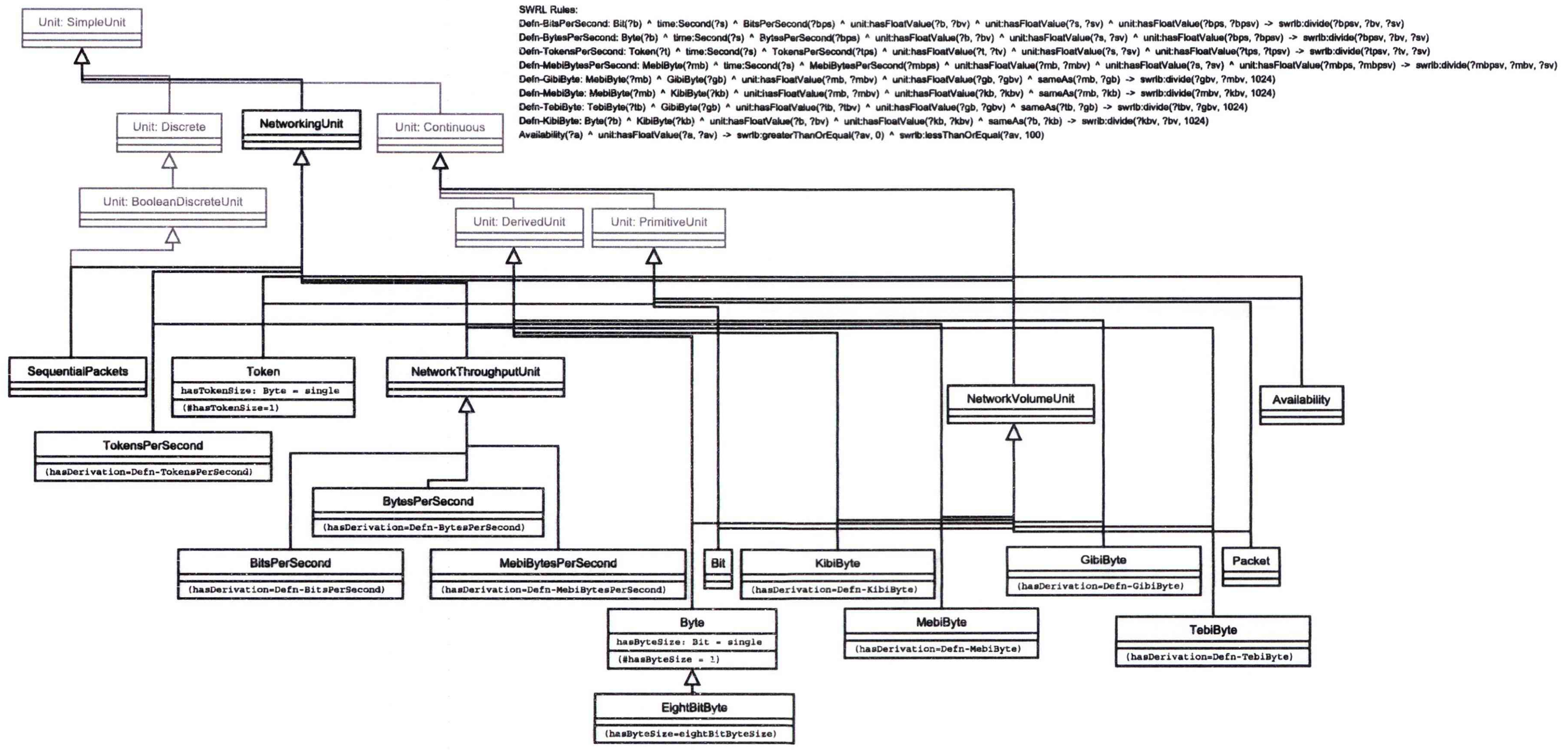
Figure 22: The Time Unit Ontology

4.4.8 Network Unit Ontology

The Network Unit ontology extends the Unit ontology in the domain of telecommunication networks. The Network Unit ontology defines a new subclass of `SimpleUnit` named `NetworkUnit` and a set of child classes of `NetworkUnit` to describe a range of units specific to telecommunication network services. Most network units are continuous units describing measurable network connectivity service parameters such as network traffic volume and throughput.

The Network Unit ontology is outlined in Figure 23. Three example Network Units are described below:

- *Throughput* describes the amount of data sent across a connection in a given period of time. Throughput is a derived unit, involving a unit of time and a unit of network traffic volume. Three expressions for network throughput are included in the Network Unit ontology. `BitsPerSecond`, `BytesPerSecond` and `MebiBytesPerSecond`. These units are not a definitive list of units for network throughput, and may be extended by users of the ontology.
- *Per Flow Sequence Preservation* is a descriptive unit for a network flow that indicates if packets in the flow transit the network in their original sequence. This parameter is important for jitter sensitive traffic where an out of sequence packet may cause a large delay in transmission. One source of out of sequence packets is load balancing at the packet level and not at the flow level. This attribute of a network flow is conveyed by the boolean discrete unit `SequentialPackets`. The value for this unit must be one of `True` or `False`, as restricted in the parent class `BooleanDiscreteUnit`.
- *Availability* may be applicable to many services such as a network connectivity service or an application level service. Availability is the average percentage of time a service is available over some time interval. Availability may be expressed as a continuous unit specified as a percentage [0,100] and represented by the `Availability` Unit in the Network Metrics ontology.


Unit: SimpleUnit
Unit: Discrete
NetworkingUnit
Unit: Continuous
Unit: BooleanDiscreteUnit
Unit: DerivedUnit
Unit: PrimitiveUnit
SequentialPackets
Token
hasTokenSize: Byte = single
(#hasTokenSize=1)
NetworkThroughputUnit
NetworkVolumeUnit
Availability
TokensPerSecond
(hasDerivation=Defn-TokensPerSecond)
BytesPerSecond
(hasDerivation=Defn-BytesPerSecond)
BitsPerSecond
(hasDerivation=Defn-BitsPerSecond)
MebiBytesPerSecond
(hasDerivation=Defn-MebiBytesPerSecond)
Bit
KibiByte
(hasDerivation=Defn-KibiByte)
GibiByte
(hasDerivation=Defn-GibiByte)
Packet
Byte
hasByteSize: Bit = single
(#hasByteSize = 1)
MebiByte
(hasDerivation=Defn-MebiByte)
TebiByte
(hasDerivation=Defn-TebiByte)
EightBitByte
(hasByteSize=eightBitByteSize)

```
SWRL Rules:
Defn-BitsPerSecond: Bit(?b) ^ time:Second(?s) ^ BitsPerSecond(?bps) ^ unit:hasFloatValue(?b, ?bv) ^ unit:hasFloatValue(?s, ?sv) ^ unit:hasFloatValue(?bps, ?bpsv) -> swrlb:divide(?bpsv, ?bv, ?sv)
Defn-BytesPerSecond: Byte(?b) ^ time:Second(?s) ^ BytesPerSecond(?bps) ^ unit:hasFloatValue(?b, ?bv) ^ unit:hasFloatValue(?s, ?sv) ^ unit:hasFloatValue(?bps, ?bpsv) -> swrlb:divide(?bpsv, ?bv, ?sv)
Defn-TokensPerSecond: Token(?t) ^ time:Second(?s) ^ TokensPerSecond(?tps) ^ unit:hasFloatValue(?t, ?tv) ^ unit:hasFloatValue(?s, ?sv) ^ unit:hasFloatValue(?tps, ?tpsv) -> swrlb:divide(?tpsv, ?tv, ?sv)
Defn-MebiBytesPerSecond: MebiByte(?mb) ^ time:Second(?s) ^ MebiBytesPerSecond(?mbps) ^ unit:hasFloatValue(?mb, ?mbv) ^ unit:hasFloatValue(?s, ?sv) ^ unit:hasFloatValue(?mbps, ?mbpsv) -> swrlb:divide(?mbpsv, ?mbv, ?sv)
Defn-GibiByte: MebiByte(?mb) ^ GibiByte(?gb) ^ unit:hasFloatValue(?mb, ?mbv) ^ unit:hasFloatValue(?gb, ?gbv) ^ sameAs(?mb, ?gb) -> swrlb:divide(?gbv, ?mbv, 1024)
Defn-MebiByte: MebiByte(?mb) ^ KibiByte(?kb) ^ unit:hasFloatValue(?mb, ?mbv) ^ unit:hasFloatValue(?kb, ?kbv) ^ sameAs(?mb, ?kb) -> swrlb:divide(?mbv, ?kbv, 1024)
Defn-TebiByte: TebiByte(?tb) ^ GibiByte(?gb) ^ unit:hasFloatValue(?tb, ?tbv) ^ unit:hasFloatValue(?gb, ?gbv) ^ sameAs(?tb, ?gb) -> swrlb:divide(?tbv, ?gbv, 1024)
Defn-KibiByte: Byte(?b) ^ KibiByte(?kb) ^ unit:hasFloatValue(?b, ?bv) ^ unit:hasFloatValue(?kb, ?kbv) ^ sameAs(?b, ?kb) -> swrlb:divide(?kbv, ?bv, 1024)
Availability(?a) ^ unit:hasFloatValue(?a, ?av) -> swrlb:greaterThanOrEqual(?av, 0) ^ swrlb:lessThanOrEqual(?av, 100)
```

Figure 23: The Network Unit Ontology

understand how to process all concepts from its adopted ontologies, however it does not necessarily need to accept the use of all concepts in any possible way.

The below example outlines the concept of ontology adoption:

- Customer A wishes to use a service provided by Provider B.
- Customer A has previously adopted ontologies P,S,U and X.
- Provider B has adopted ontologies O,P,U,V and X.
- A and B discover each other's adopted ontologies. The mechanism by which this occurs is not specified as it is out of scope.
- A resulting SLA can be expressed using common ontologies: P,U and X, and both parties should understand all concepts from these ontologies, however A or B may not agree to the use of a concept in a particular way.

4.5.2 SLA Individual Creation

Service Level Agreement individuals are created from the `SLA` class and are considered complete when all of the OWL constraints applicable to the SLA are satisfied. For example, the `SLA` class has a restriction defined on the `hasConsumer` property, stating it must refer to exactly one OWL individual. An `SLA` individual is only complete once an OWL individual has been assigned to the `hasConsumer` slot of the SLA.

4.5.3 SLA Validation

A Service Level Agreements is a contracted pair of commitments for some service. As described above, SLA individuals have constraints which must be satisfied for the SLA to be valid. An SLA individual may be submitted to some validation method to check its conformance to the stated constraints. Validation may be performed by one or more signatories on an SLA before the SLA is agreed to.

A valid SLA individual should be syntactically correct and satisfy all appropriate constraints defined for the utilised concepts. OWL syntax and constraint validators are freely available such as the vOWLidator[12].

Additionally, the ontologies used to describe the SLA concepts should be consistent and satisfiable. Consistency checking ensures that an ontology does not contain any contradictory facts. Creating an individual from an unsatisfiable class will cause the ontology to be in an inconsistent state. Consistency of an ontology, or combination of ontologies can be checked with a consistency checker, commonly implemented as a

12 http://owl.bbn.com/validator/

component of knowledge reasoners. Many reasoners are available for OWL, with some such as Hoolet[122] offering support for limited SWRL rule sets.

4.6 Summary

Within this chapter, a set of ontologies were described that can be used as a generic base for describing secure service level agreements with formal semantics. To maintain flexibility, the SLA description is kept independent from any techniques used to manipulate the SLAs. Examples and suggestions for use are provided to guide the use of the ontologies.

An ontology of temporal concepts is included that has a natural relationship with the event calculus, allowing temporal reasoning over created SLAs. To enable the delivery of a service in accordance with an SLA to be verified, an interface is defined between the service parameters expressed on an SLA and corresponding real-world measurements. Finally, the SLA structure has a service specification (functional service information) that is separate from the contractual information (non-functional information), allowing the use of a generic SLA structure for different services.

Chapter 5
The QDINE Negotiation Framework

5.1 Introduction

This chapter describes the QDINE negotiation framework that supports the automated and ubiquitous delivery of service level agreements to fixed or mobile users of telecommunications networks. The delivery of these service level agreements addresses a key component of the comprehensive service delivery platform, introduced in chapter 1.

Chapter 3 outlines the requirements of such an SLA negotiation framework. These requirements are satisfied by the QDINE negotiation framework described herein.

By adopting the QDINE framework, end-users are free to roam within the constraints of possible network connectivity. Services are provided to users on request and these services may be handed over to alternative, more suitable providers as required. An end-user is responsible for negotiating service contracts autonomously between itself and service providers, and is supported by a means to verify its identity as a basis for trust and to establish accountability within untrusted networks. Services are automatically billed to users via ad-hoc billing paths.

The QDINE service delivery universe is divided into multiple individual service markets. Each *market* is a multiagent system $\{\alpha, \beta_1 \ldots \beta_n, \gamma_1 \ldots \gamma_t, \delta_1 \ldots \delta_p, \epsilon_1 \ldots \epsilon_s\}$ containing one market agent α (§5.6.3) and multiple service providers β (§5.6.2) wishing to participate in that market. Each market may have multiple subscribers ϵ (§5.6.4) receiving information about services in the market and multiple consumers γ (§5.6.1) requesting services from that market. Each consumer may nominate a billing provider δ (§5.6.5) to manage service billing and verify its identity to other players. Communication between agents is described in (§5.2).

Market size is the number of consumers accessing services from registered providers in that market. The QDINE framework does not place a limit on market size, which is determined by a combination of the accessible user base, constraints enforced by the market agent and constraints or incentives placed by the service providers. For network connectivity services, the market size will likely be determined by a combination of the technology domain size, administrative domain size, geographic scope and political boundaries among other factors.

In line with the requirement of autonomy (§3.1.2), the framework is designed so that each player has autonomous authorisation for distribution of resources controlled by that player. That is, every player has ultimate responsibility for its own resources. A consumer's resources are its funds, for a service provider, it is the services provided, and a billing provider's, resources are its reputation and the provided proxy financial responsibility on behalf of a contracted user. Agents acting in the market agent or subscriber roles possess no resources of concern to the QDINE framework.

Each entity involved in the QDINE framework is an autonomous agent and may be a human interacting via an appropriate interface or an electronic agent. Agents are responsible for managing interactions with other agents as required to achieve the delivery of services from provider to consumer. Agents within the QDINE framework are assumed to be self-interested; that is, they act to maximise their own utility and goals and so this is a competitive multiagent system. Agents are autonomous and as such can not be enforced to act in a particular way, however their actions may be guided by incentives and permissions granted by others. Any agent may participate in one or more markets.

Following the scope of the QDINE framework as outlined in sections 5.1.1 and 5.1.2 below, section 5.2 discusses the way in which agents in a QDINE system communicate. Section 5.3 describes how the behaviour of agents in the framework may be influenced, and the impact of inappropriate behaviour. Section 5.4 continues with a discussion of the processes involved in a QDINE market; including service discovery, negotiation and renegotiation for a roaming user. All the standard QDINE interactions that occur between agents are described in section 5.5. Section 5.6 describes in detail the roles played by agents in a QDINE market; including their capabilities, norms and rights they can grant to others. Appendix B specifies the QDINE-specific vocabulary used by agents in the framework, such as the possible actions and statements.

5.1.1 Goals

The goal of the QDINE framework is to provide a flexible, secure SLA negotiation framework in accordance with the requirements stated in chapter 3. In creating the framework, the following goals were defined.

- Define the format of messages sent between agents.

- Define a language to express the content of exchanged messages.
- Define a language for describing interaction protocols used by agents.
- Define a set of dialogues for agents to perform tasks using the QDINE framework.
- Define the roles involved in a QDINE multi agent system.
- Define a set of agent roles and their responsibilities in the framework.
- Allow agents to discover billing methods supported by other agents in the market.

5.1.2 Out of Scope

The QDINE framework does not:

- Model the internal behaviour of agents such as whether a BDI model or lightweight agent approach is used. The programming language is also not prescribed. The participating agents may be human.
- Describe methods to determine the reputation of other agents.
- Enforce the norms of interactions between agents.
- Describe a finite list of protocols available to agents negotiating in the framework.
- Specify a billing method used by agents.

5.2 Communication within QDINE

Within the QDINE framework, an agent communicates with one or more recipients using message payload μ with content χ. χ is expressed in a metalanguage[13] λ, using concepts from ontologies $\omega_1 \dots \omega_t$. μ has one performative ϕ defining the purpose of the communicative act. μ may be part of conversation ψ using interaction protocol ρ.

The QDINE framework uses the FIPA Message Transport System (MTS) for message transport as it is a standardised, agent-based message transport designed to promote interoperability between agent platforms. Communication may occur between agents on one platform, or between agents from different platforms. Every message sent in the QDINE framework is required to comply with the FIPA MTS standard and consists of two components (Figure 24):

- An envelope header containing mandatory components such as transport level addresses, date and the ACL representation used, as well as optional fields such as encryption details.
- The message payload (μ) expressed in the FIPA Agent Communication Language (FIPA-ACL)[174].

13 A *metalanguage* defines the way in which concepts are expressed in some concrete form. For example, KIF, Java, and First Order Logic are all metalanguages as they describe the way concepts are presented, not the concepts themselves.

Attribute	Description of Attribute
`performative` (ϕ)	Purpose of utterance
`Sender`	Participant in communication
`receiver`	Participant in communication
`reply-to`	Participant in communication
`content` (χ)	Content of message (OWL-CL)
`language` (λ)	Content metalanguage
`encoding`	Content encoding
`ontology` ($\omega_1 \ldots \omega_t$)	Vocabularies used within content
`protocol` (ρ)	Name of current protocol
`conversation-id` (ψ)	Unique ID of current conversation
`reply-with`	Desired ID of reply message
`in-reply-to`	Unique ID of reply message
`reply-by`	Time limit for reply

Table 2: FIPA-ACL Message Payload Attributes

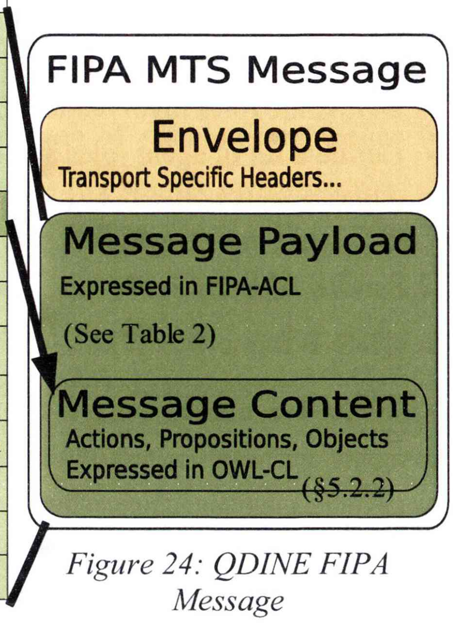


Figure 24: QDINE FIPA Message

5.2.1 The QDINE Message Payload

Table 2 lists the standard FIPA-ACL message payload attributes; the names of which are standardised by FIPA. Each attribute describes a slot in a FIPA-ACL compliant message payload. The FIPA-ACL standard requires every message payload to specify one performative, all other parameters are optional.

The QDINE framework places constraints on some ACL message payload slots. Each slot from Table 2 is described below, along with its constraints as required by the QDINE framework. A sample QDINE message payload is illustrated in Example 11.

For the sake of readability, the message payload is simply referred to as the "message", as the FIPA MTS message and envelope are transport concerns and not important to this work.

```
(request
:receiver (set (agent-identifier :name wilma))
:sender (agent-identifier :name fred)
:content

~~<?xml version="1.0"?>~~
~~<rdf:RDF~~
~~  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"~~
~~  xmlns:owl="http://www.w3.org/2002/07/owl#"~~
~~  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"~~
~~  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"~~
~~  xmlns:owl-cl="http://qdine.it.uts.edu.au/ontologies/OWL-CL.owl#"~~

~~  xmlns="qdine://agent1@some.agent.platform:3709/JADE#"~~
~~  xml:base="qdine://agent1@some.agent.platform:3709/JADE#">~~
~~  xmlns:qint="http://qdine.it.uts.edu.au/ontologies/QDINE_Interaction.owl#"~~

<owl-cl:MessageContent>
  <owl-cl:hasAction
    rdf:resource="http://mytel.com/kb/usr/C34.owl#DefaultRegistrationAction"/>
</owl-cl:MessageContent>

:language OWL-CL
:protocol
  http://qdine.it.uts.edu.au/ontologies/protocols.owl#consumerRegistration
:ontology qint=http://qdine.it.uts.edu.au/ontologies/QDINE_Interaction.owl#
:conversation-id C001324978604T0
:reply-with R009
:reply-by 1163404851)
```

```
*1 This standard OWL text may be
   omitted because it is implied
   by the use of OWL-CL.

*2 Within the QDINE framework, the
   URIs of the base and default
   namespaces are assumed as per
   §5.2.1.3.1. Additionally, the
   namespace listings may be moved
            to the :ontology slot.
```

Example 11: An Example QDINE Message

```
<rdf:RDF xmlns="http://mytel.com/kb/users/C034561.owl#"
  ... (other namespace declarations)
  ...>
  <owl:Ontology rdf:about="">
    <owl:imports
      rdf:resource="http://qdine/ontologies/QDINE_Interaction.owl"/>
  </owl:Ontology>
  <qint:ProxyAuthentication rdf:ID="Credentials">
    <qint:usingIdentifier
      rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
        C34
    </qint:usingIdentifier>
    <qint:authenticatedBy rdf:resource="http://mytel.com/kb/auth.owl#AAID"/>
  </qint:ProxyAuthentication>
  <qint:AuthoriseConsumer rdf:ID="DefaultAuthorisationAction">
    <qint:bill-to rdf:resource="http://mytel.com/kb/biller#BPAID"/>
    <qint:usingCredentials rdf:resource="#Credentials"/>
  </qint:AuthoriseConsumer>
  <qint:RegisterConsumer rdf:ID="DefaultRegistrationAction">
    <qint:understoodOntology
      rdf:resource="http://qdine/ontologies/QDINE_ViolationOntology.owl"/>
    <qint:understoodOntology
      rdf:resource="http://qdine/ontologies/TimeUnitOntology.owl"/>
    <qint:understoodOntology
      rdf:resource="http://qdine.it.uts.edu.au/ontologies/QDINE_SLA.owl"/>
    <qint:supportedNegotiationMechanism
      rdf:resource="http://qdine/ontologies/negMechanisms.owl#TwoPass"/>
    <qint:includingServiceSubscription
      rdf:resource="#DefaultSubscriptionAction"/>
    <qint:includingAuthorisation rdf:resource="#DefaultAuthorisationAction"/>
  </qint:RegisterConsumer>
  <qint:ProvideServiceUpdates rdf:ID="DefaultSubscriptionAction"/>
</rdf:RDF>
```

Example 12: The supporting OWL file for Example 11. This file would be published online

The `performative` of a message describes the intention of the sender in making an utterance. The QDINE framework adopts the standard performatives defined by FIPA for FIPA-ACL. These performatives are described in detail in [175], which includes the requirements when using each performative. The data type for the contents of the `performative` slot is String.

Within QDINE, two often used performatives are `request` and `inform`. The content of a request message must include an action for the recipient to perform. When sending a request message, the sender of the message desires that the receiver(s) perform the described action. The contents of an inform message is a propositional statement. The sender of the message believes the receiver(s) do not have the knowledge represented in the propositional statement and the sender wishes for this knowledge to be shared.

5.2.1.2 Sender and Receiver

Every message may include the agent id of the `sender` and one or more `receivers`. Within QDINE, there is no requirement to specify the sender or receiver of a message. If omitted, the transport level address contained in the message envelope will be used. The data type of the contents for both the `sender` and `receiver` slots in a QDINE message is String.

5.2.1.3 Content

The `content` of a message contains the primary information intended to be communicated. The message content is described in a particular metalanguage, using a vocabulary from one or more ontologies. The content of a QDINE message is expressed using OWL-CL(§5.2.2), and is subject to the following constraints:

- OWL-CL allows the formalisation and transmission of knowledge described using OWL. OWL resources included within OWL-CL content may be anonymous or they may be identified by some unique name within an XML namespace. If an agent intends to refer to a sent resource in future messages, that resource must be identified. A message receiver is not required to store any identified resources received, but may choose to do so. If a resource is referenced in future messages and the receiver has not stored the resource, the receiver may return a "not understood" message to the sender, with a reason of `unknownRDFResource` (§B.2.8.14), returning the IDs of any unknown resources.
- OWL-CL allows the transmission of knowledge. At any point, the acceptance of knowledge described on a received message may cause the receiver's knowledge

base to become inconsistent. The way in which an agent handles such a situation is not strictly prescribed, however some strategies are discussed in section 5.7.1.

- Within QDINE, OWL-CL content may exclude the `<?xml version="1.0"?>` header tag required for RDF 0.9 compliance. Additionally, the opening and closing `<rdf:RDF...>` tags may also be omitted including the list of namespaces in use. If these tags are omitted, the namespace/prefix mapping is achieved as follows:
  The `xml:base` and RDF default namespace are assumed to be the QDINE agent specific namespace(§5.2.1.3.1) of the message sender. All other namespaces in use should be listed in the `ontology` slot in the QDINE message. If a message recipient requires the RDF tags and namespaces in the message `content`, and they are not sent, a "not understood" message may be returned with the reason of "strict OWL required"(§B.2.8.3).

5.2.1.3.1 Content Scope

OWL resources used within OWL-CL message content in the QDINE framework may be of two types; QDINE specific or globally accessible. Global resources are standard RDF resources, not specific to the QDINE framework. These resources are assumed to be semi-permanent such as an ontology definition file hosted online. Global resources may be of any type expressible in RDF or OWL such as Class, Property or Individual. QDINE specific resources are created for use within an interaction between two or more agents. They are therefore temporary in nature and exist only for the sender and receivers of the message on which they are contained. Because of their dynamic nature, QDINE specific resources are restricted to the set of OWL Individuals, and may not be new Classes or Properties. If an agent were allowed to create OWL Classes or Properties as QDINE specific resources, the receiving agent may not understand or know how to process the new concepts.

Every RDF resource is represented within some namespace. The namespace of a QDINE specific resource is composed by appending a scheme prefix of "`qdine://`" to the agent name of the message sender. In this way, a QDINE specific resource may be distinguished from a regular RDF resource by its scheme prefix. The standard namespace suffix "#" is also included. An example QDINE specific namespace is given in Figure 25.

```
qdine://agent1@some.agent.platform:3709/JADE#
```

Figure 25: An example QDINE resource namespace

5.2.1.4 Language

The FIPA-ACL `language` slot specifies the metalanguage used to form the message `content`. The data type of the `language` slot contents in a QDINE message is String. The `language` slot of all QDINE messages is required to be populated with the string "`OWL-CL`", identifying the use of the OWL-CL content language(§5.2.2)

5.2.1.5 Ontology

The `ontology` slot is used to indicate the ontologies that define the vocabulary used within the message content. To participate in a QDINE market, the minimum vocabulary required for an agent is that defined by the QDINE Interaction ontology (Appendix B). Additionally, concepts from any other ontology may be used in a message content.

Within QDINE, message content is sent using OWL-CL. As such, the ontologies used may be specified directly in the namespace listing within the opening `<rdf:RDF>` tag (The struck-out text in Example 11). However, within QDINE, a message sender may choose to omit the opening `<rdf:RDF>` tag, in which case the ontology namespaces are listed in the `ontology` slot, as defined below:

The contents of the `ontology` slot in a QDINE message is required to be in the following format, described in EBNF:

```
ontology contents ::= prefix, "=", namespace,  ("#" | "/")
                      [gap-separator, ontology contents].
```

The term `gap-separator` is defined in the ISO EBNF specification[14]. A `namespace` is the unique identifier of the ontology being described. The format required for a `namespace` is the standard URI format[15], without any trailing solidus ("/") character. The `prefix` is a shortened ontology identifier that is unique within the message content. Within the message content, the prefix is attached to XML tags to identify the ontology that they were drawn from. The format of the `prefix` term used in the above EBNF definition is the standard XML namespace prefix[16].

One or more of the ontology listings may be omitted if the receiving party is assumed to know the namespace to prefix mappings. Omitting an ontology listing breaks the

14 ISO_IEC_14977

15 http://www.w3.org/Addressing/URL/5_URI_BNF.html

16 http://www.w3.org/TR/xml-names/#NT-Prefix

assurance of a common understanding of concepts used, and should only be done if a strong assumption can be made that each party knows the correct namespace to prefix mapping.

If an unlisted prefix is used within the message content and that prefix is unknown to the recipient, then a "not understood" message may be returned to the sender with a reason of "unknown namespace prefix"(§B.2.8.1), listing the prefix(es) unknown. Additionally, if an ontology is specified, which the receiver does not support, a "not understood" message may be returned with a reason of "unsupported namespace"(§B. 2.8.2), listing all the ontology namespaces used that are not supported.

5.2.1.6 Encoding

The transmission of a message requires that it be serialised using some encoding. Accordingly, the FIPA ACL message `encoding` slot refers to the way the content of a message is serialised into binary form. A typical encoding is UTF-8. If an encoding for the contents isn't specified, the encoding of the message envelope is assumed to apply.

5.2.1.7 Protocol

An agent utterance exists as part of an interaction sequence. The interaction may be a single communicative act, or may form part of a large, multi-round interaction protocol. The `protocol` slot can be used to identify a protocol, ρ ; describing the rules that govern an interaction sequence. The data type of the contents of the `protocol` slot is String. Messages are not required to indicate a protocol in use, however it is advised for multi-message dialogues when expectations are placed on participants.

All QDINE-specific protocols, such as provider or consumer registration(§5.5.2.2, §5.5.2.1) are described using the Protocol Description Language (PDL)(§5.2.3). For QDINE specific protocols, the `protocol` slot must be populated with the URI of the PDL `Protocol` instance that specifies the protocol in use.

Non QDINE-specific protocols, such as those used for SLA negotiation are not required to use PDL. In such case, the protocol slot should be populated with a string that unambiguously identifies the protocol in use.

5.2.1.8 Conversation-id, reply-to, reply-with, in-reply-to and reply-by

In addition to the `protocol` slot, A FIPA-ACL message has five slots used to control agent dialogue. If a protocol is specified in a message, the current instantiation of that protocol must be identified through the use of the `conversation-id` slot. If the sender of a message desires a reply be sent to a different agent address, the `reply-to` slot can be populated with an agent-id that is different to the sender. To identify a reply to a particular utterance, an agent can populate the `reply-with` slot with some identifying string, σ. On reply to such a message, the replying agent will populate the `in-reply-to` slot of the reply message with σ. In the QDINE framework, if a message sender requires a reply to an utterance by a specific time, τ, the `reply-by` slot of the sent ACL message is populated with τ, represented as an epoch time; that is, the number of milliseconds since January 1, 1970, 00:00:00 GMT.

5.2.2 OWL-CL

Within FIPA, the term *content language* is used to describe both a metalanguage, and a core content vocabulary required to support the standard FIPA performatives. As such, the `language` slot of a message describes not only a metalanguage, but also indicates an available core vocabulary for the message content.

A new content language named OWL-CL has been defined as part of the QDINE framework. The metalanguage for OWL-CL is the semantic web ontology language (OWL). Message content described using OWL-CL is standard OWL content and as such can be parsed by any OWL parser. OWL-CL is not specific to the QDINE framework, and may be used with any any FIPA compliant multiagent system desiring an ontological content language. A requirement of using OWL-CL is that the sender of a message using OWL-CL content is required to adopt the xml1.0 standard for the content, and understand the xml[17], rdf[18], rdfs[19] and owl[20] namespaces, as well as the OWL-CL namespace.

The root component of any OWL-CL message is a `MessageContent` individual (§5.2.2.1), to which all other content in the message is associated via properties defined on the `MessageContent` class. As with all OWL classes, the `MessageContent` class may be extended to suit a particular purpose.

It may be necessary to include one message payload within the content of another message payload, such is the case with the FIPA `proxy` performative[175]. Doing so

17 xml="http://www.w3.org/XML/1998/namespace"

18 rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"

19 rdfs="http://www.w3.org/2000/01/rdf-schema#"

20 owl="http://www.w3.org/2002/07/owl#"

will mean there is more than one individual of the `MessageContent` class within an OWL-CL `content` slot. Typically, the structure of an XML document can be analysed to determine a root element, however OWL documents are largely structure independent, and in such a case of multiple `MessageContent` instances, it may not be possible to know which is the root `MessageContent` individual. The below method is defined to determine the root message content:

> Every message with content specified in OWL-CL must include exactly one OWL individual[21], φ that is of the class `MessageContent` or a descendant class, and is not used within the range of the `hasContent` property of another individual. φ is then the root component of that message.

In addition to defining the metalanguage, OWL-CL defines a minimal ontology of concepts to describe message content as required for conformance with standard FIPA performatives. Nine basic OWL classes are defined and are loosely specified in sections 5.2.2.1-5.2.2.5 below. When specifying the classes, an approach of maximum flexibility was taken by applying minimum constraints to the created classes. This was done so that the concepts are applicable to a wide scope of application domains. As such, most classes should be extended when used within a specific domain. The QDINE Interaction ontology(Appendix B) extends the basic OWL-CL components for use specifically in the QDINE framework. Figure 26 presents the relationship between the OWL-CL classes and Figure 27 illustrates how these classes may be used within OWL-CL message content.

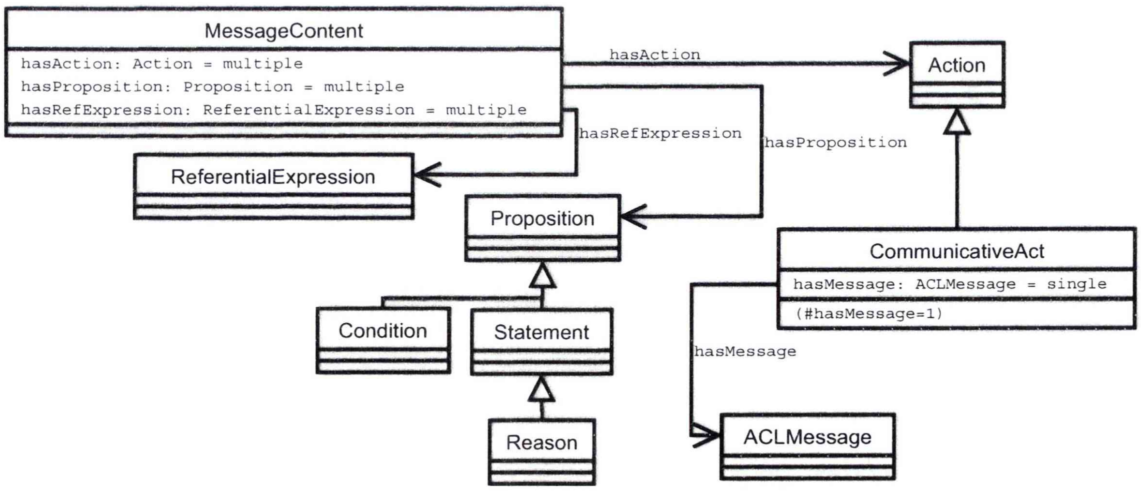
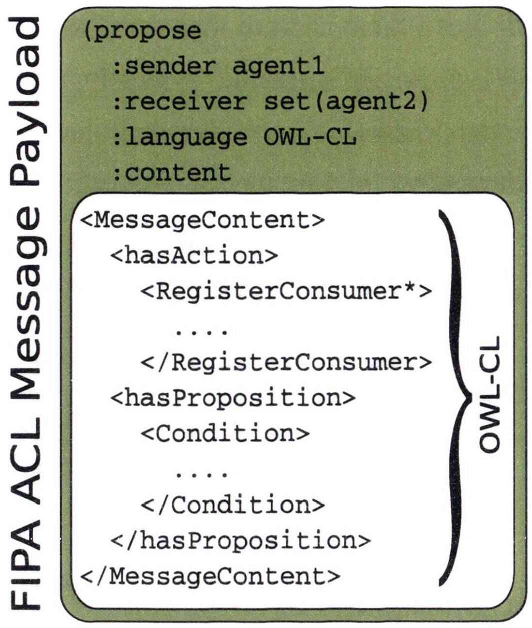


Figure 26: OWL-CL class diagram

21 OWL concepts are discussed further in section 2.7.2.1

5.2.2.1 MessageContent

The `MessageContent` class encapsulates the entire content slot of a message. Three properties of the `MessageContent` class have been defined for conformance with the FIPA standard performatives. These properties are `hasAction`, `has-Proposition` and `hasRef-Expression`, the domains of which are `Action`, `Proposition` and `ReferentialExpression` respectively. Additional properties can be applied to `MessageContent` subclasses as required on implementation.



```
(propose
  :sender agent1
  :receiver set(agent2)
  :language OWL-CL
  :content
<MessageContent>
  <hasAction>
    <RegisterConsumer*>
      ....
    </RegisterConsumer>
  <hasProposition>
    <Condition>
      ....
    </Condition>
  </hasProposition>
</MessageContent>
```

* `RegisterConsumer` is a sub-class of `Action`

Figure 27: Example ACL Message using OWL-CL

5.2.2.2 Action

An action represents some activity that an agent performs.

5.2.2.2.1 CommunicativeAct

A communicative act is the action of sending a message. In addition to the properties the `Action` parent class, the `CommunicativeAct` class has one extra property: `hasMessage` that indicates the message to be sent as part of the action.

5.2.2.3 Proposition

A proposition is a statement that can be either true or false. Three proposition subclasses are defined.

5.2.2.3.1 Condition

A condition is a proposition that the sender requires to be held true by the receiver.

5.2.2.3.2 Statement

A statement is an informational proposition. It is an assertive statement of some knowledge by the sender to the receiver.

Reason
A reason is a informational proposition describing why something happened.

5.2.2.4 ReferentialExpression

The `ReferentialExpression` message component describes a set of possibly incomplete individuals, the identity of whom may be unknown. A referential expression is often used in queries, where the intention is for the recipient to bind the unknown set to valid and complete individuals.

5.2.2.5 ACLMessage

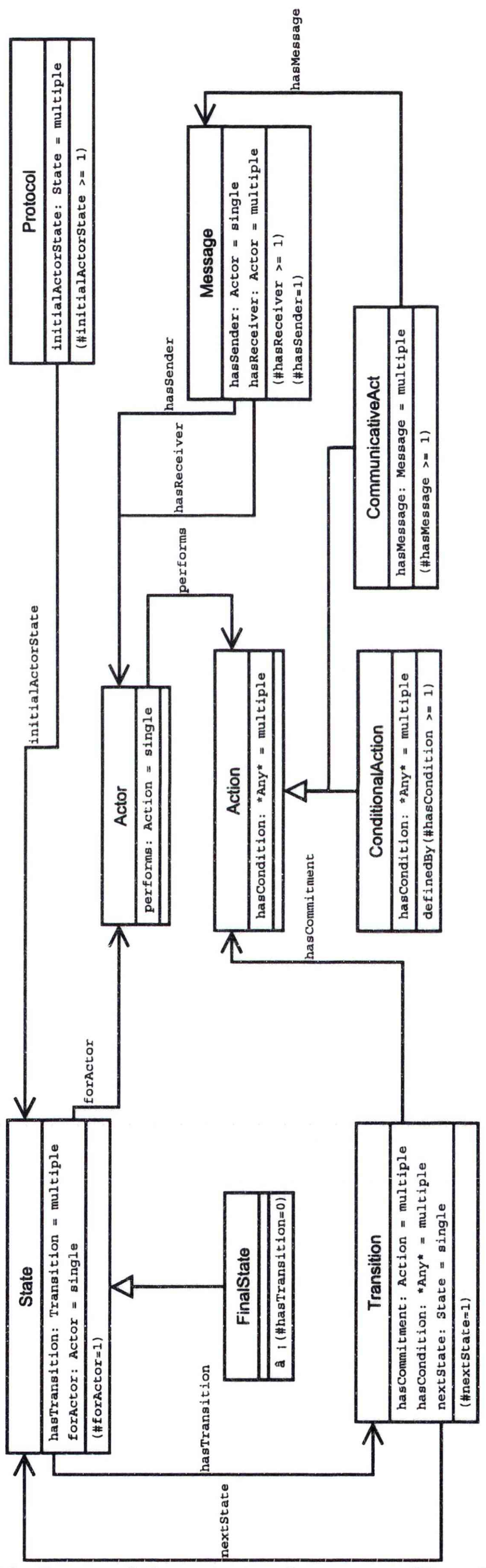
An `ACLMessage` is an object used to describe an entire ACL message payload for inclusion within some other message content as part of a communicative act. An example use for this class is for the FIPA "proxy" performative, that requires the content to include a communicative act, holding a message to send to the final destination.

5.2.3 Protocol Description Language

To provide a formal definition of the protocols used within the QDINE framework, a new state-based *Protocol Description Language* (PDL) is defined. PDL is based on conditional state-transition logic with optional commitments on transition. PDL is a generic protocol description language and as such is not only applicable to QDINE, but may be used for modelling interaction protocols in any domain.

PDL is described using an OWL ontology to maintain an ontological approach. PDL consists of modified classes from the DAML interaction protocol ontology [176], with extensions for commitments and formal protocol transition conditions and actions along the lines of OWL-P[177], but with consistent semantics based around a state / transition mechanism. The classes, relationships and constraints of the Protocol Description Language are presented in Figure 28.

Every `Protocol` described using PDL specifies one or more initial actor states in the protocol. A *state* defines an external representation of an actor's current activity. Each state may have one or more transitions leading to other states in the protocol.


Protocol
initialActorState: State = multiple
(#initialActorState >= 1)
Message
hasSender: Actor = single
hasReceiver: Actor = multiple
(#hasReceiver >= 1)
(#hasSender=1)
hasMessage
hasSender
hasReceiver
CommunicativeAct
hasMessage: Message = multiple
(#hasMessage >= 1)
performs
initialActorState
Actor
performs: Action = single
Action
hasCondition: *Any* = multiple
ConditionalAction
hasCondition: *Any* = multiple
definedBy(#hasCondition >= 1)
hasCommitment
forActor
State
hasTransition: Transition = multiple
forActor: Actor = single
(#forActor=1)
FinalState
(#hasTransition=0)
Transition
hasCommitment: Action = multiple
hasCondition: *Any* = multiple
nextState: State = single
(#nextState=1)
hasTransition
nextState
Figure 28: Protocol Description Language Class Diagram

PDL does not model states of a protocol, but states of the actors within a protocol. This approach allows a more flexible protocol description than conventional tools such as UML sequence diagrams. This is because strict temporal synchronisation between actors is not required. If desired, actor states can be synchronised through bound message passing. This is achieved by adding a "send message" commitment on the message sender's transition and a "received message" constraint on the receiver's transition, and specifying the sent and received message as the same message individual.

Communication protocols necessarily involve the sending and receiving of messages. One subclass of action is therefore defined as a communicative act. The `CommunicativeAct` class includes a slot for a transmitted message. A generic `Message` class is also defined to be used with a communicative act. A `Message` specifies the sending and (possibly multiple) receiving actors.

All the QDINE specific interaction dialogues are specified using PDL, and are outlined in section 5.5.

5.2.3.1 Transitions

Transitions are the key component of a PDL protocol as they define the progression of actors within the protocol. A transition may only be defined from state π_i to π_j given both states define a common actor. Every `Transition` must include a target state for the actor and may include the following optional components, describing the norms of a protocol:

- One or more publicised conditions that must all hold for the transition to occur. Conditions on transitions define what is necessary before a transition can occur. These conditions may be expressed in any constraint language such as SWRL.
- One or more commitments that the role adopts when the transition occurs. Commitments on a transition are specified as actions which the transiting role promises to perform. An `Action` provides a formal description of some activity an agent performs. Any action may have restricting conditions which must hold before the action can begin, and must hold while the action is being performed. Such an action is a `ConditionalAction`.

Attaching one or more conditions to a transition does not mean that whenever the conditions hold, the transition must occur. It means the transition can *only* occur *if* the conditions hold. An actor may also employ private conditions which must also hold for a transition during a particular state to occur. These private conditions are not modelled in the protocol, for they are implementation specific and are not required to be advertised to other participants. Private conditions are discussed here to illustrate why public conditions on a transition may hold but the transition may not be triggered. These private conditions form the *trigger* for the transition. The flow of a transition is thus:

Conditions hold + trigger occurs → Commitments adopted → Move to next state

5.2.4 Communication Security

Communication between players should be secure. Authentication, message integrity, privacy and accountability are addressed by the QDINE framework.

5.2.4.1 Authentication

Borrowing from Internet Single Sign On (SSO) technologies[178][179], the QDINE framework uses the following authentication model, based on asymmetric encryption[180][181] and the Public Key Infrastructure[182]. An agent is authenticated by the inclusion of its identifying credentials on a sent message. A receiver can then validate those credential using the key validation protocol(§5.5.2.9). Within QDINE, two authentication credentials are used:

1. An X.509[183] digital certificate, signed by a trusted Certification Authority (CA), or
2. QDINE Proxy Authentication

CA signed digital certificate credentials are used for millions of transaction daily on the Internet and requires the authenticated entity to purchase a digital certificate from some trusted certification authority such as Verisign[22]. The certificate is a statement by the certification authority that it authenticates the identity of the entity listed on the certificate, subject to it holding the private key that corresponds to the public key on the certificate.

Any QDINE agent can use traditional CA signed digital certificates within QDINE. The QDINE Proxy Authentication method was developed for the following reasons:

- CA signed digital certificates can be expensive. Small service providers or end users should not be expected to bear such a cost to participate in the system.
- CA Signed digital certificates offer an additional administrative overhead to the identified entity. An aim of the QDINE project is to reduce administrative load for the participants.

22 http://www.verisign.com/

- Within QDINE, a primary reason for identifying a consumer is so the correct entity is billed for financial charges accumulated. The billing provider (§5.6.5) is ideal to authenticate the contracted agent as it is responsible for the financial decisions made by the agent, and can quickly revoke the authentication for an irresponsible agent. Additionally, the billing entity will already have sufficient details about the authenticated entity to certify its identity without any additional administrative overhead.
- Billing authorisation provided by a billing provider to a service provider is given for a particular digital certificate. If a billing provider wants to assign different expiry times to different billing authorisations, it can create different certificates for the authorisations. Such flexibility is infeasible using traditional CA signed digital certificates.

5.2.4.1.1 QDINE Proxy Authentication

The QDINE Proxy Authentication uses digital certificates as the identifying credential, however the certificates are managed and signed by a billing provider who takes responsibility for authenticating the identity of an agent. The proxy authentication mechanism bases the authentication of a contracted agent on the trust of the associated billing provider. The authentication process occurs as follows.

1. The user generates a public and private key pair and registers the public key with a Billing Provider. This step happens once and may be performed manually by the user via a web interface on the billing provider's website or by another off-line means. When registering a public key for some customer id, the billing provider must verify the identity of the registering user. In addition, the billing provider must ensure the user has the matching private key for the registered public key. This can be tested through a key challenge protocol such as the key validation protocol(§5.5.2.9).

   The system relies on an agent's private key remaining known only to that agent. If the privacy of that key is breached, the user must generate a new key pair and register the new public key with the billing provider, revoking the previous public key.

2. When an agent ς wants to be identified to another agent, it can attach a Proxy Authentication object to a sent message. The Proxy Authentication object includes the address of the billing provider to contact for a certificate and a customer id of ς at that billing provider.

3. The agent receiving the proxy authentication object can authenticate the sending

agent by first contacting the billing provider for a certificate for the specified customer ID. This is done via the Agent Authentication dialogue (§5.5.2.4).

4. The returned certificate containing a public key for the user and signed by the billing provider can be used for the key validation protocol (§5.5.2.9) to ensure the user has the corresponding private key for the certificate.

Digital Certificates remain valid throughout a specific time period. All entities in a QDINE market should therefore have synchronised clocks within a broad tolerance to ensure correct authentication. This requirement is easily satisfiable using the existing and widely used network time protocol [184].

5.2.4.2 Message Privacy

Messages passed between agents in the system may travel through untrusted networks. Payment details or other sensitive information may require confidentiality within communicating agents. Public Key Encryption is used to secure sensitive messages from unintended recipients.

Encryption may occur at the socket layer via SSL or TLS. Additionally, the content of ACL messages may be encrypted. A security add-on for ACL called X-security has been developed [185] allowing encryption of ACL message content using public key cryptography. Additionally, as the content language used in the QDINE project is based on XML, the standard XML encryption methods [186] may be used.

5.2.4.3 Accountability

Service Level Agreements are an electronic contract for a service. To ensure that these contracts are not disputable, the interactions should be traceable and positively identify the sender.

All agents in the framework possess a private and public encryption key. Messages sent between agents can be signed with the sender's private key. In this way, a sent message can be provably attributed to a particular private key and hence, to one entity.

Section 4.3.2 outlines the repudiation issues with messages based on Semantic Web technologies. Within QDINE, non-repudiable message content is achieved by instantiating components as `NonRepudiableThings` (§4.4.2.1) which ensures a common ontology version is used for all parties on an SLA.

5.2.4.4 Integrity

Ensuring the integrity of a message involves proving that the message has not been tampered with after being sent. A common method of achieving this is by building a secure digest of a message and signing it with the sender's private key.

A secure digest is unique to the message used to build the digest. If the message is altered in any way, a different digest will be built by application of the digest algorithm on the altered message. One of the SHA-2[187][169] family of algorithms may be used to generate a secure digest. To date, these algorithms remain uncompromised and are recommended for standardised secure signing of data.

The X-Security add-on for ACL may be used to attach signature and digest information to an ACL message. Additionally, message content may employ the XML digital signature framework [188].

5.2.4.5 SLA Signatures

A service level agreement may be digitally signed to provide accountability of the parties to the agreement and verify SLA data integrity. A digitally signed object may contain the signature internally such as in Figure 29, or the signature may be specified externally to the signed object as in Figure 30. Signature schemes such as xmldsig[188] allow both forms of signature, however internal signatures introduce large complications to prevent the signature data from impacting the signature value. This is because the Embed and Extract operations must be lossless. That is, the output from an Embed, followed by an Extract operation must be exactly the same as the input to the Embed operation.

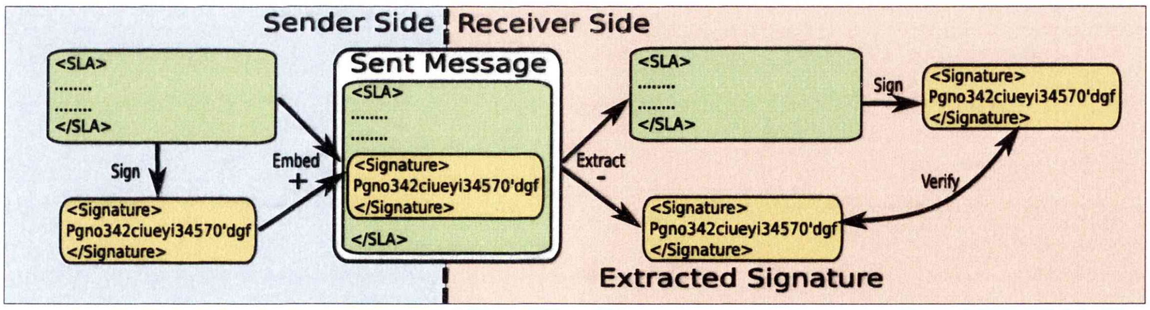


Figure 29: Internal SLA signature

Alternatively, an external signature does not require complex embed or extract operations, thereby providing lower computational overhead and fewer opportunities for error. QDINE does not prescribe the use of any particular signature method, but suggests the use of external SLA signatures.

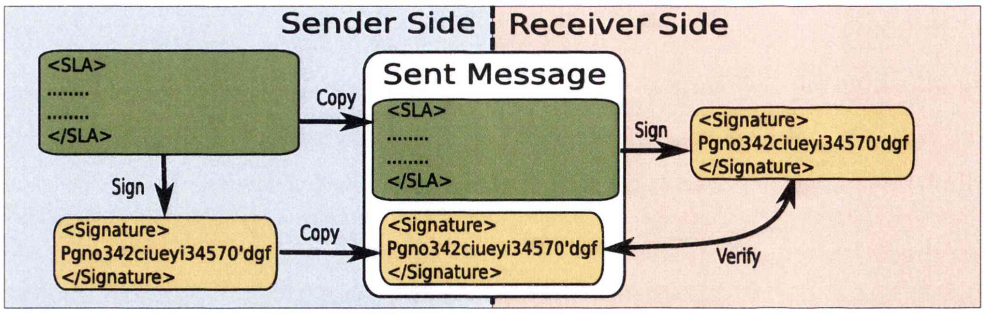


Figure 30: External SLA Signature

To ensure accountability and non-repudiation for SLAs based on the semantic web, a signed SLA must be non-repudiably associated to a prefix→ namespace map for all the XML tag prefixes used within the SLA. For example, if the tag `<c3:price>` is used on an SLA, the mapping from prefix `c3` to the correct namespace URI must be associated to the SLA, otherwise a signatory to the agreement could claim the concept `price` was assumed to be from a different namespace, with a different meaning.

One method to associate prefix mappings to an SLA is to include the mappings in the same signed block as the SLA. In that way, the signature value encompasses both the SLA and prefix mappings. Within QDINE, the entire FIPA message `content` slot can be used as the signed block for an SLA. If a well formed OWL document is used within the `content` slot, both the SLA to be signed, and all the prefix mappings for the ontologies used in the SLA are included in the signed block.

5.3 Agent Behaviour

Agents within the QDINE framework are assumed to be self-interested; that is, they act to maximise their own utility and goals and so this is a competitive multiagent system. Agent actions are driven by their goals and may be guided by incentives and permissions granted by others. Agents are autonomous and as such can not be enforced to act in a particular way. They are expected to act in accordance to the norms of a market, but correct agent behaviour is not assumed.

Adherence to market norms is enforced in a peer to peer fashion by the individual participants. This is the approach taken with dialogue games [189] and maintains strong flexibility in agent interaction. In recent years, the concept of Electronic Institutions (EI) [190] has appeared, where agent interactions are policed by an administrative body. The EI concept enables powerful role/dialogue validation and is a natural progression for multiagent systems where strict validation of agent interaction is required. If desired, the norms of service negotiation protocols may be enforced by the market agent, as is performed within electronic institutions. For this to occur, all protocol messages must pass through the market agent. Potentially, in future QDINE work, the market agent may operate an electronic institution to host the service negotiation protocols.

5.3.1 Market Norms

Norms represent obligations, rights, responsibilities, or other normative concepts of an agent [191]. As such, norms describe an agent's expected behaviour. There are three QDINE specific situations where norms are asserted over an agent's behaviour:

- Protocol Norms – Protocol norms define the valid sequence of messages, along with any conditions and commitments involved with the interaction. The Protocol Description Language(§5.2.3) can be used to define normative interaction protocols.
- Role Norms – When participating in a QDINE market as a particular role, there are norms associated with that role within the market. Section 5.6 describes the norms applicable to the roles in a QDINE market.
- Commitment Norms – Commitment norms are norms governing the enactment of a commitment. The primary aim of the QDINE framework is to establish service level agreements between consumers and providers. These service level agreements are a commitment by the signed parties to act in accordance with the obligations stated on the SLA.

5.3.2 Violation of Norms

Violation of norms may occur, and affected agents should be designed to handle such occurrences. The way in which an agent responds to the violation of a norm by another agent is not prescribed by QDINE, however some techniques are available by which an agent can respond to a norm violation:

- Withdrawal of Rights – An agent acting in a market role has a set of rights which it can grant to other roles in the market. An agent may withdraw these rights from another player in response to violation of a norm.
- SLA Violation – If a violation model is specified within an SLA obligation (§4.4.3.2), the model may be automatically evaluated to determine the impact on the violating party.
- Exclusion – Any agent may deny communication with another agent who has violated a market norm.
- Impact on Reputation – As described in section 5.3.3, a reputation model may be built about the agents within a QDINE market. Norm violation may result in the modification of an agents reputation score by the affected agents.

- Legal Action – Strong, non-repudiable commitments may be made by agents, such as a statement of billing authorisation or a signed service level agreement. If the impact on a player from a commitment violation is large, and the outcome for such a violation is not previously agreed, as is done with Service level Agreements, the affected parties may pursue legal action with the violators.

5.3.3 Trust and Reputation

Trust is concerned with the maintenance of expectation. Establishing trust in another player involves verifying their identity and ascertaining their probability of satisfying or exceeding expectation. Correct identification ensures a player with which you believe you are interacting is in fact that player. Given the correct identification of players, an attempt to model their trust can be made.

A prior probability of a player performing to expectation may be determined from their reputation, built from the outcome of one or more past interactions between players. REGRET[192], a reputational model for trust, approaches reputation from three dimensions: Individual, Social and Ontological. Multiple aspects of reputation may be combined by a player to form a view of another.

- The Individual dimension of reputation models the direct interaction between two players. When the same two players have repeated interactions, private trust models may be built about opponents.
- In situations where one-off interactions between players are common, Social reputation becomes important as players may have no previous private history on an opponent. In this situation, membership with a group may contribute to an opponent's reputability. Alternatively, a collaborative reputation model may be consulted, built from the outcomes of multiple individual interactions.
- As an alternative to a single reputability score, the ontological dimension of reputation divides reputability into separate aspects, each applicable to a domain of concern. For example, an agent may have a good reputation in selling bikes, but a bad reputation for telling the correct time. The agent's bad "time" reputation may not affect your decision on purchasing a bike from that agent.

The QDINE framework deals with the identification aspect of trust, but does not attempt to model or handle reputation. Players within the system are securely identified, but no assumptions are made concerning discovery of the reputation of others. Distributed reputation management systems are available[193][194][195] and such a system may be adopted to manage reputation between agents in a QDINE market. The integration of reputation management with QDINE is left for future work.

If the reputability of an agent can be ascertained, an opponent can use this information to grant rights and privileges to that agent for interactions within a QDINE market.

5.4 The QDINE Market

A QDINE market is a multiagent system $\{\alpha, \beta_1 \ldots \beta_n, \gamma_1 \ldots \gamma_t, \delta_1 \ldots \delta_p, \epsilon_1 \ldots \epsilon_s\}$ containing multiple agents; each adopting one or more market roles. The QDINE market roles are discussed in detail in section 5.6. Every QDINE market has exactly one agent α acting in the market agent role(§5.6.3). A market agent defines the service market and as such, there can only be one agent performing the market agent role per service market.

Each market can have multiple agents adopting the service provider role β (§5.6.2) providing services to that market. Additionally, each market may have multiple subscribers ϵ (§5.6.4) receiving information about services in the market, as well as multiple consumers γ (§5.6.1) requesting services from that market. Each consumer may nominate a billing provider δ (§5.6.5) to manage service billing and verify its identity to other players.

Any agent participating in a QDINE market may adopt multiple roles and an agent may participate in multiple markets. Figure 31 (pg. 120) outlines a sample QDINE market. Within Figure 31, the direction of the solid arrows indicates the initiator of that interaction.

Agents within a QDINE market are involved with five main processes to manage the life-cycle of service level agreements between consumer and provider: Service Publication, Service Discovery, Consumer Authorisation, SLA Negotiation, Service Handover, and SLA Termination. Agent interactions that are required to perform these processes are executed using the QDINE agent dialogues (§5.5) over the QDINE communication system, previously described in section 5.2.

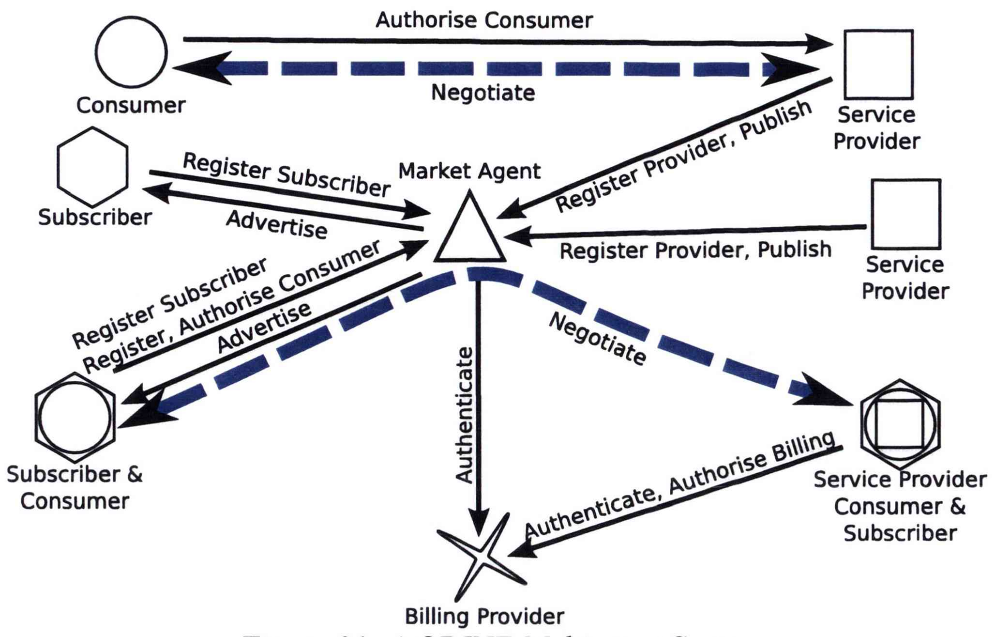


Figure 31: A QDINE Multiagent System

5.4.1 Service Publication

The service publication function is concerned with the creation of a distributed directory of available services, from which the consumers can then choose for their individual requirements.

The service publication function operates as follows:

1. A real world entity creates an agent, α such that $\alpha \in A \subseteq Z$ where A is the set of all of agents performing the QDINE Market Agent role, and Z is the set of all QDINE capable agents connected to one or more telecommunication networks.

2. The address of α is distributed to agents, $\zeta_1 \ldots \zeta_n \subseteq Z$ via some information distribution mechanism. Network protocols such as DHCP may be adopted, or the information may be distributed via some manual means. The way in which this occurs is not a concern of the QDINE framework.

3. One or more agents, $\beta_1 \ldots \beta_m \subseteq \zeta_1 \ldots \zeta_n$ register with α, such that $\beta_1 \ldots \beta_m \subseteq B \subseteq Z$ where B is the set of all agents performing the role of a Service Provider.

4. $\beta_1 \ldots \beta_m$ publish templates for their available services $\kappa_{\beta_{1n}} \ldots \kappa_{\beta_{mn}}$ to α. Formally, $\kappa_{\alpha_1} \ldots \kappa_{\alpha_n} \subseteq \kappa_{\beta_{1n}} \ldots \kappa_{\beta_{mn}} \subseteq K$ where K is the set of all QDINE services, $\kappa_{\beta_{1n}} \ldots \kappa_{\beta_{mn}}$ are all the services available from providers $\beta_1 \ldots \beta_m$, and $\kappa_{\alpha_1} \ldots \kappa_{\alpha_n}$ is the set of all services published at market agent α.

5.4.2 Ad-hoc Service Discovery

The ad-hoc service discovery process enables fixed and roaming consumers to discover services available via their connected networks.

1. A QDINE agent ζ roams within the physical world and may be connected at times to one or more telecommunication networks through a network access device. As ζ roams between networks, it discovers the addresses of market agents $\alpha_1 \ldots \alpha_n$ that are available via ζ's connected networks. The way in which the market agents are discovered is undefined and not a concern of the QDINE project.
2. ζ may choose to register with a market agent $\alpha \in \alpha_1 \ldots \alpha_n$ as subscriber ϵ, such that $\epsilon \in E \subseteq Z$ where E is the set of all agents acting in the QDINE Subscriber role. On registration as a subscriber, α advertises its available services to ϵ.
3. Whenever the services available to α change through publication from local service providers, α may propagate the changes to ϵ. Constraints on the propagation strategy are defined in the norms of the market agent role (§5.6.3).

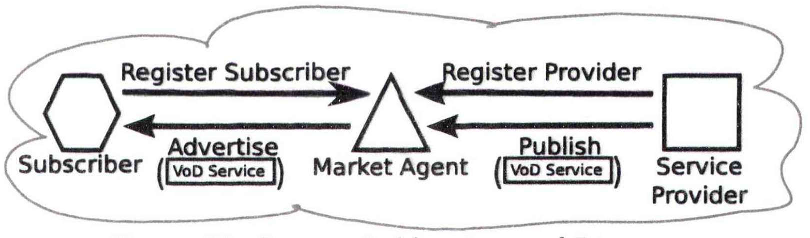


Figure 32: Service Publication and Discovery

5.4.3 Consumer Authorisation

The Consumer Authorisation process allows the market agent and service providers in a market to authorise some agent as a consumer for services. It is expected that most, if not all service providers will require an agent to be authorised as a consumer before being allowed to negotiate for services.

1. Before a QDINE agent ζ can participating in negotiations for service level agreements from a provider β, β may require that ζ is authorised with β as consumer γ such that $\gamma \in \Gamma \subseteq Z$, where Γ is the set of all agents acting in a QDINE Consumer role.
2. If the negotiation process for a service desired by a consumer involves a market agent α, α may require that ζ is registered and authorised with α as a consumer γ. On ζ's registration as a consumer with α, α may forward the authorisation request to all services providers $\beta_1 \ldots \beta_m$, removing the need for a consumer to

request authorisation directly from every provider.

3. Before β authorises ζ as a consumer, β may require billing authorisation from ζ's nominated billing provider δ. Additionally, β may require an established billing path between β and δ.

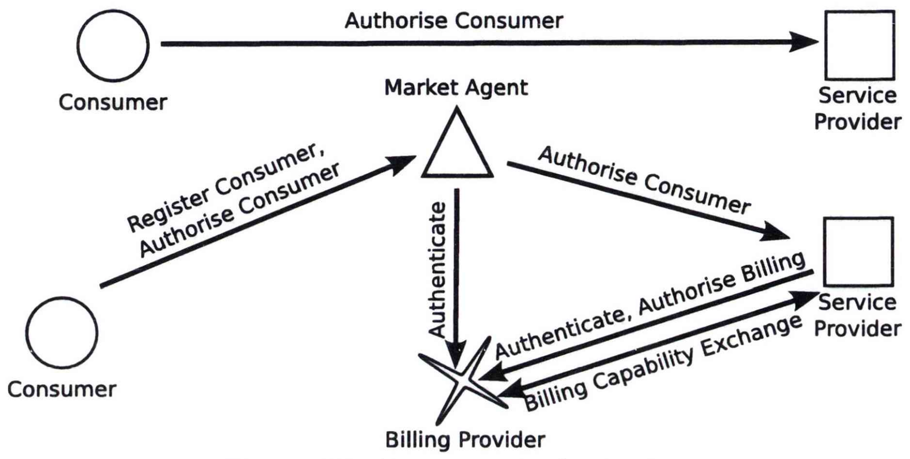


Figure 33: Consumer Authorisation

5.4.4 SLA Negotiation

The goal of any negotiation mechanism used in the QDINE framework is to create a mutually accepted valid instance of an SLA based on a set of shared ontologies between parties.

Previous to negotiating for an SLA, the consumer has been verified to the service provider as a customer of a particular billing provider, and possibly authorised by the market agent to participate in the negotiation. SLAs can then be negotiated with the provider confident the agreements will be honoured by the consumer.

Depending on the type of service provided and market properties, such as service configurability or level of competition, different negotiation models may be appropriate. For example, a reverse auction may be appropriate when the goods are configurable and there are multiple providers. Alternatively, a posted price market may be appropriate for fixed goods with a well known value. A bilateral bargaining, argumentation-based model may be appropriate with multiple attribute goods and a single provider, when convergence time for negotiation is not important.

Negotiation within the QDINE framework is therefore not limited to a single mechanism. Any mechanism may be used and is indicated on the service publication and by the protocol identifier specified in the initial negotiation utterance. A service negotiation mechanism may include the market agent, one or more consumers and one or more providers. The billing provider or subscriber market roles are not involved in

SLA negotiation. Some example negotiation mechanisms are discussed in section 5.5.2.11.

At the completion of every negotiation mechanism, an SLA binding process is performed. This binding process ensures the consumer and provider have a non-repudiable commitment to the SLA negotiated.

5.4.4.1 SLA Binding

The SLA binding process consists of three message exchanges. These messages may be

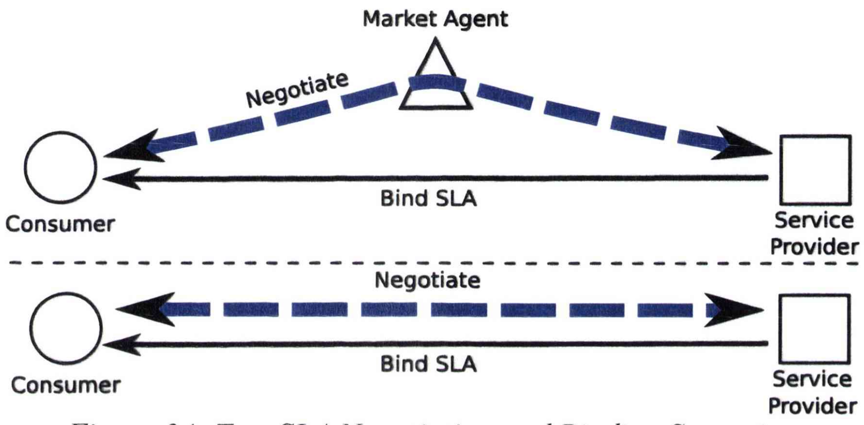


Figure 34: Two SLA Negotiation and Binding Scenarios

included in the final stages of a negotiation mechanism, or be performed after a negotiation mechanism is complete. It is assumed that by the time the SLA binding process is performed, the consumer and provider have agreed on the SLA to be committed. Figure 35 describes the binding process.

The first message; the service provider's proposal to provide a service is conditionally binding, based on the consumer accepting the proposal within a given timeout T_n. If the consumer agrees with the proposal, an acceptance is returned. This acceptance is conditional on the consumer receiving an acknowledgement of the acceptance within the timeout T_n. If the proposal is not accepted, the consumer sends a rejection and the interaction sequence is terminated.

Once the provider sends an acknowledgement, the consumer is obliged to the service if it does not send a cancel message by the timeout time. If the timeout time elapses and the provider does not receive a cancel message, the provider is legally obliged to provide the described service, and the consumer can expect the service will be available. At timeout elapse, the consumer is also bound to the SLA and any cancellation penalties are then applicable.

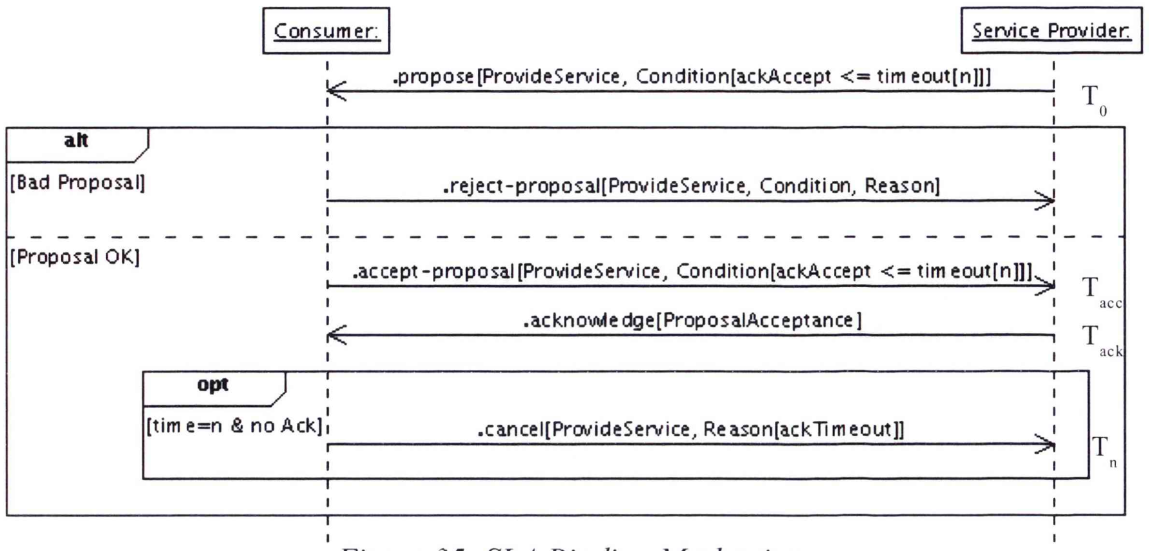

Consumer:
Service Provider:
.propose[ProvideService, Condition[ackAccept <= timeout[n]]]
T0
alt
[Bad Proposal]
.reject-proposal[ProvideService, Condition, Reason]
[Proposal OK]
.accept-proposal[ProvideService, Condition[ackAccept <= timeout[n]]]
Tacc
.acknowledge[ProposalAcceptance]
Tack
opt
[time=n & no Ack]
.cancel[ProvideService, Reason[ackTimeout]]
Tn

Figure 35: SLA Binding Mechanism

5.4.4.2 SLA Termination

As discussed in section 4.4.3.4, the possible SLA termination scenarios are dependant on the service being provided. The model for SLA termination is therefore assumed to be described within an SLA specification, and not addressed within the negotiation framework.

5.4.5 Service Pre-negotiation for Fast Handover

Once an SLA has been agreed, it may be desired, or necessary for continued service to hand the service over to a different provider. A handover between identical service types is known as a *horizontal handover*. An example of a horizontal handover is the process that occurs as a mobile phone subscriber roams between transmitting base stations. Additionally, for session-based services (§2.1.3) a session established on one type of service may be transferred to a new session on a different type of service. Such a handover between different service types is known as a *vertical handover*. To illustrate concepts within this section, the following vertical handover example is used:

Example 13: A mobile user has initiated a telephone voice call via his 3G mobile handset to a friend on a fixed telephone line. The mobile user roams into a WIFI zone where he transfers the active call across to a Skype-out session over the WIFI network. Figure 38 (pg. 129) illustrates this scenario.

In addition to the vertical/horizontal division, a handover may occur in a hard or soft handover environment:

- In a hard handover environment, a handover is *required* to ensure continuation of service. An example here is when a consumer physically moves location and is forced to change network connectivity, thereby making an existing service provider no longer available.
- In a soft handover environment, a handover is *desired* by the consumer because a different available service offers some advantage over the existing service. An example here is when a cheaper or otherwise preferred service provider and/or service type becomes available which can perform the required function.

Handover of a service or service session in a ubiquitous service environment (§2.6) involves the technological aspect of the handover, and may also require the creation or adjustment of business relationships with the source and/or target providers. A target provider may require an established business relationship with the customer and an SLA for the new service before accepting the technological handover. The QDINE framework does not address the process of performing the technological service handover; however the advertisement of a provider's technical handover capabilities, and management of the business relationships is addressed.

5.4.5.1 Handover Capabilities

The following method is made available by QDINE to inform consumers of the technical handover capabilities that are available for a given service.

There are three types of entity that may be involved in a service handover:

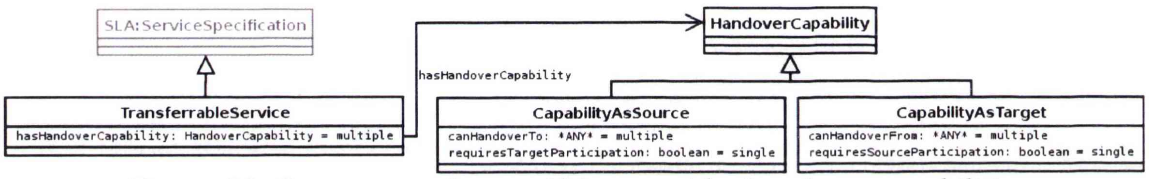
- The consumer initiating the handover,
- The source and target service providers, and
- One or more remote peers interacting in the service with the handover-initiating consumer.

To perform a service handover, any subset of the entities involved with either the source or target service may be required to facilitate the technical handover process. Before a consumer can attempt a handover, it will ensure that all required parties can support the handover. The following method enables the consumer to discover the possibilities for handover from a currently active service; based on the handover capabilities of the involved parties.

Within QDINE, the initiating consumer is aware of its own handover capabilities, and must discover the handover capabilities of any remote peers if they are required to participate in the handover process. The way the initiating consumer discovers its peers' handover capabilities is not addressed here, and left for future work (§7.2). The service handover capabilities of a provider, including any constraints placed on those handovers are described on the published service specifications, as outlined below.

5.4.5.1.1 Provider Handover Capabilities

To describe its handover capabilities, a service provider may publish its services using the `TransferableService` class. A `TransferableService` includes the `hasHandoverCapability` property, which can be used to specify one or more handover capabilities. The `HandoverCapability` class defines the provider capabilities in supporting a handover, as well as the requirements of the handover. Each `HandoverCapability` may be further defined as a `CapabilityAsTarget` or `CapabilityAsSource`.



SLA:ServiceSpecification
HandoverCapability
hasHandoverCapability
TransferrableService
hasHandoverCapability: HandoverCapability = multiple
CapabilityAsSource
canHandoverTo: *ANY* = multiple
requiresTargetParticipation: boolean = single
CapabilityAsTarget
canHandoverFrom: *ANY* = multiple
requiresSourceParticipation: boolean = single

Figure 36: Components for specifying provider handover capabilities

A `CapabilityAsTarget` indicates a provider's ability to manage a handover *from* any of the specified services. The property `canHandoverFrom` property is used to list the services from which the target provider can manage a service handover. If a `CapabilityAsTarget` also has the `requiresSourceParticipation` property set to `true`, then the source services must must explicitly define a `CapabilityAsSource` supporting an outgoing handover to the target service. If the `requiresSourceParticipation` flag is not set, or set to false, the target provider indicates its capability to manage the handover without indication of support from the source provider. This requirement is expressed formally below:

$$\exists ?ts, ?ct, ?ss\ .\ \mathit{hasHandoverCapability}(?ts, ?ct) \wedge \mathit{canHandoverFrom}(?ct, ?ss) \\ \wedge\ \mathit{requiresSourceParticipation}(?ts, \mathit{true}) \Rightarrow \exists ?cs\ .\ \mathit{hasHandoverCapability}(?ss, ?cs) \wedge \\ \mathit{canHandoverTo}(?cs, ?ts)$$

Correspondingly, a `CapabilityAsSource` indicates a provider's ability to manage a handover *to* any of the specified services. A `CapabilityAsSource` includes the property `canHandoverTo`, listing the services to which it can manage a service handover. If a `CapabilityAsSource` also has the `requiresTarget-Participation` property set to `true`, then the target services must must explicitly define a `CapabilityAsTarget` supporting an incoming handover from the source service. If the `requiresTargetParticipation` flag is not set, or set to false, the source provider indicates its capability to manage the handover without indication of support from the target provider. The above participation requirement is expressed by the following rule:

$$\exists ?ss, ?cs, ?ts \,.\, \mathit{hasHandoverCapability}(?ss, ?cs) \wedge \mathit{canHandoverTo}(?cs, ?ts) \\ \wedge\ \mathit{requiresTargetParticipation}(?cs, true) \Rightarrow \exists ?ct \,.\, \mathit{hasHandoverCapability}(?ts, ?ct) \wedge \\ \mathit{canHandoverFrom}(?ct, ?ss)$$

To illustrate the concept of provider handover capabilities, Example 13 from page 124 is continued. The 3G voice service which the user originally subscribed to is defined as a transferable service, and includes a `CapabilityAsSource` with a `can-HandoverTo` property specifying the Skype-out application. This Skype-out handover capability does not require target participation. An OWL description for this capability is given below:

```
<qint:TransferableService rdf:ID=UMTSVoice>
  <qint:hasHandoverCapability>
    <qint:CapabilityAsSource>
      <qint:canHandoverTo rdf:resource="http://skype.com/defns/SkypeOut/>
    </qint:CapabilityAsSource>
  </qint:hasHandoverCapability>
</qint:TransferableService>
```

5.4.5.2 The Pre-negotiation Process

To fulfil the handover requirements from the business perspective, the Consumer Authorisation (§5.4.3) and Service Negotiation (§5.4.4) processes are performed with target provider, and the current SLA modified, or a new SLA created for the target service. Additionally, the service agreement with the source provider may be terminated.

- In a soft handover environment, the consumer has direct access to the source and target service providers, so can manage the required processes directly and establish service level agreements with the target providers using the standard negotiation mechanisms.
- In a hard handover environment, there will likely be insufficient time for both the required business processes *and* the technical handover to be performed when both the source and target providers are directly available to the consumer,.

To allow seamless handover in a hard handover environment, a technique is described for pre-emptive negotiation (or pre-negotiation) of service contracts before a technical handover is required. Using pre-negotiation, only the technical handover is performed during the service availability overlap. The following pre-negotiation process is an extension of the standard QDINE processes (§5.4.1 - §5.4.4).

Service Publication....

1. Any service specification κ, published to a market agent α may be defined as both pre-negotiable and transferable κ_{pt}, where in addition to the service specific attributes and the properties inherited from a transferable service (§5.4.5.1.1), a pre-negotiable service includes the agent ID of the originating market agent; that is, the agent ID of α, where the service was originally published.

2. On receipt of κ_{pt} by α, if the agent ID of α is not included on κ_{pt} as the originating market agent, then α is required to add this information to the service description.

Service Discovery....

1. A market agent α_1 discovers its neighbouring market agents $\alpha_j...\alpha_m$ through some information distribution mechanism. This mechanism by which this occurs is not defined by QDINE.

2. α_1 may register as a subscriber ϵ_α with neighbouring market agents $\alpha_j...\alpha_m$ to discover the services available in those markets. Services available from $\alpha_j...\alpha_m$ are advertised to ϵ_α following the standard service discovery process (§5.4.2). A subset of these services may be pre-negotiable services.

3. When α_1 advertises its available services to its subscribers, the list of services advertised may include any subset of the pre-negotiable services α_1 has discovered from neighbouring market agents $\alpha_j...\alpha_m$, as well as any of the services published by the local providers.

4. $\gamma\epsilon_1$ is a consumer and subscriber with α_1 and is continuously informed of the local services available and their handover capabilities, as well as neighbouring services and their handover capabilities. $\gamma\epsilon_1$ can use this information to generate handover possibilities for any active service sessions.

Consumer Authorisation & Service Negotiation....

1. At some time, $\gamma\epsilon_1$ may wish to pre-emptively negotiate a service from a neighbouring market for continuation of an active service session. To pre-negotiate the service, $\gamma\epsilon_1$ may first need to authorise itself with the desired neighbouring service provider and/or market agent, and can then negotiate for the service.

2. The neighbouring service provider and/or market agent may not be accessible by $\gamma\epsilon_1$ directly. In such a case, $\gamma\epsilon_1$ may use its local market agent, α_1 as a message proxy, brokering the interaction between $\gamma\epsilon_1$ and the neighbouring agents.

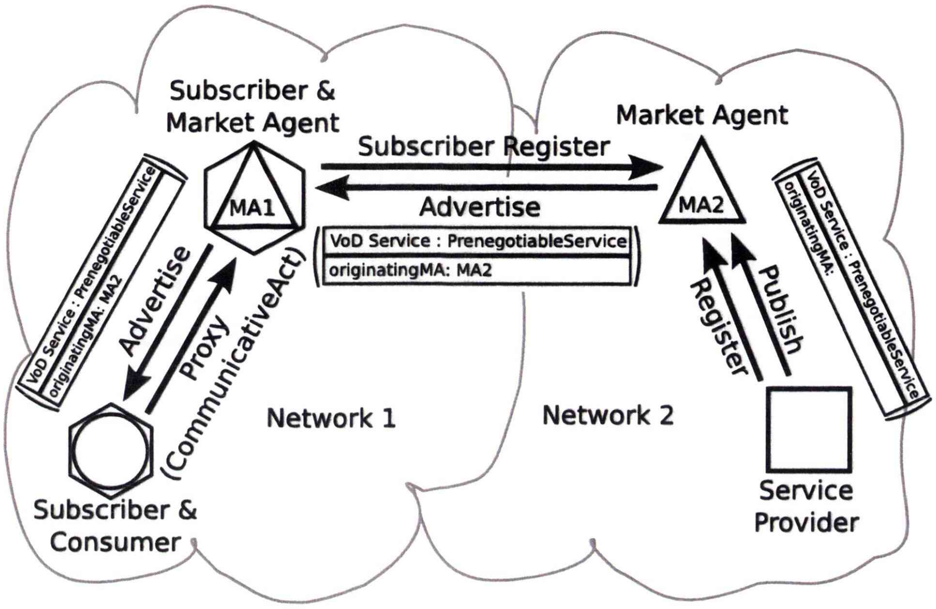


Figure 37: Service Pre-negotiation

5.4.5.3 Use Case

Example 13 is continued, and is illustrated by the scenario presented in Figure 38 and Table 3. The UMTS and WIFI networks are in a hard handover situation and each

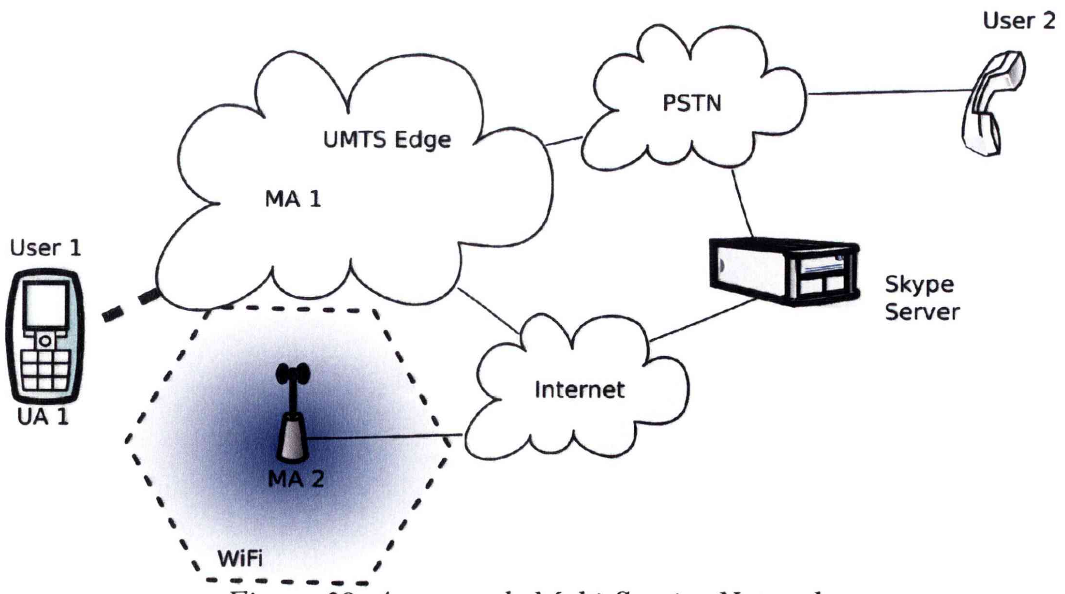


Figure 38: An example Multi-Service Network

network operator is providing a market agent for its own network (MA1 and MA2).

The handover occurs as follows:

- MA1 and MA2 discover each other's available services by becoming a subscriber in each other's markets.
- User 1 is represented by a QDINE agent UA1 and is connected to the network served by MA1. UA1 registers as a consumer and subscriber with MA1.
- MA1 advertises the local services and the QoS IP Connectivity service available from MA2 to UA1.
- UA1 subscribes to the mobile UMTS connectivity service from the Network Operator at MA1, in order to receive incoming and make outgoing voice calls.
- User 1 initiates a voice call to User 2 over the UMTS network.
- UA1 recognises it will roam into the MA2 zone, and knows the UMTS Voice service can be transferred to a Skype-out session and so pre-negotiates an appropriate IP Connectivity agreement with the Network operator at MS2, using MS1 as a message proxy.

Market Agent 1		Market Agent 2	
Service Provider	**Available Services**	**Service Provider**	**Available Services**
Network Operator	UMTS Connectivity Voice, Conference Call. SMS / MMS Mobile IP	Network Operator	QoS IP Connectivity FON
UMTS Net. Oper.	UMA		

Table 3: Service Registrations for Figure 38

- As soon as UA1 picks up the signal from the Network Operator associated with MA2, the technical handover is made, based on the pre-established SLA.

5.5 QDINE Agent Dialogues

A set of QDINE specific agent dialogues have been defined to describe the standard interactions in a QDINE framework. Interactions occur between agents, with agents participating as a particular actor in the dialogue.

There are 19 QDINE dialogues and 12 dialogue actors defined: 5 actors were named in accordance with the QDINE market roles to illustrate where dialogues are specific to market roles. 7 actors have generic names as they are used by multiple market roles. The 12 actor names are:

- Consumer
- Service Provider
- Market Agent
- Billing Provider
- Subscriber
- Key Challenger
- Key Holder
- Authorisation Initiator

- Authorisation Responder
- Authenticating Agent
- Billing Capabilities Exchange Initiator
- Billing Capabilities Exchange Responder

Agents are autonomous, and as such, an agent may send messages which are not associated with any of the QDINE dialogues. These non-QDINE messages may exist as part of an externally defined dialogue and may be indicated by the `supported-NegotiationMechanism` property within an agent registration request. Service negotiation mechanisms are not standardised by the QDINE framework, however they are a key component for distributing service level agreements. As such, some examples are presented in section 5.5.2.11.

In addition to service negotiation mechanisms, a further source of non-QDINE messages may be those required to operate a service. The use of such messages may be assumed by the use of a particular service, or indicated within a service specification.

5.5.1 Patterns of Interaction

Most QDINE interactions conform to one of three interaction patterns: Request, Inform and Acknowledged Inform. These three patterns are described below, using a sequence diagram and a state-transition diagram for each participant in the interaction. Dialogues that do not conform to the above three patterns are described with their own custom diagrams within the dialogue description.

Messages on the sequence diagrams are labelled with the performative of the message, followed by the content components of a message, enclosed within square brackets [..]. For example, the label "`.request [Action]`" indicates the `request` performative with an `Action` as the message content.

Transitions on the state diagrams are labelled with all the possible transitions from one state to another. Vertical bars "|" within the labels indicate an "or" relationship between the surrounding transitions. For example, the label "`Timeout | Receive Success`" indicated that the illustrated change of state may be caused by either the `Timeout` or `Receive Success` transitions. Each transition on a state-transition diagram typically matches to a message on the corresponding sequence diagram, however internally triggered transitions such as time-outs and silent agreements to perform an action are also labelled. These internal transitions indicate to other participants why an actor may be behaving in a particular way.

If a received message is not understood, the receiver may respond with a message using the FIPA "not understood" performative. This possibility is a feature of any message sent using FIPA ACL, and as such, these messages are not included on the protocol diagrams. When a sent message is not understood, the states of the agents is assumed to be the same as before the offending message was sent.

5.5.1.1 Request Interaction Pattern

The request interaction pattern allows an initiating agent to request that the responder perform a given action. The the two primary messages of concern are the initial request and the final success message. Correspondingly, there are two key content components: The `Action` specified in the initial request, and the `ActionDone` success proposition.

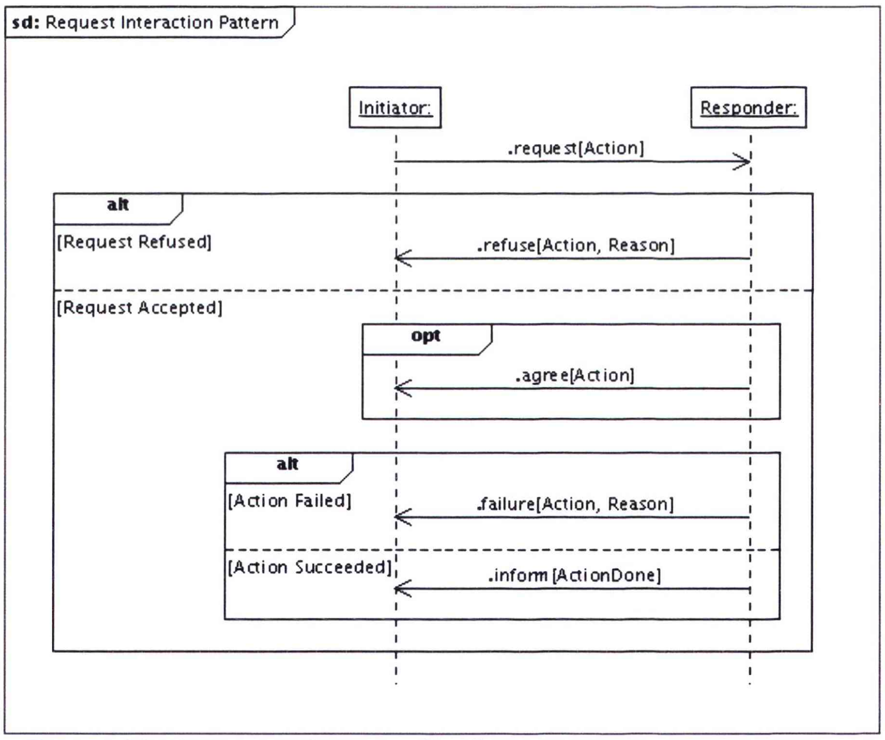


Figure 39: Sequence diagram for the Request interaction pattern

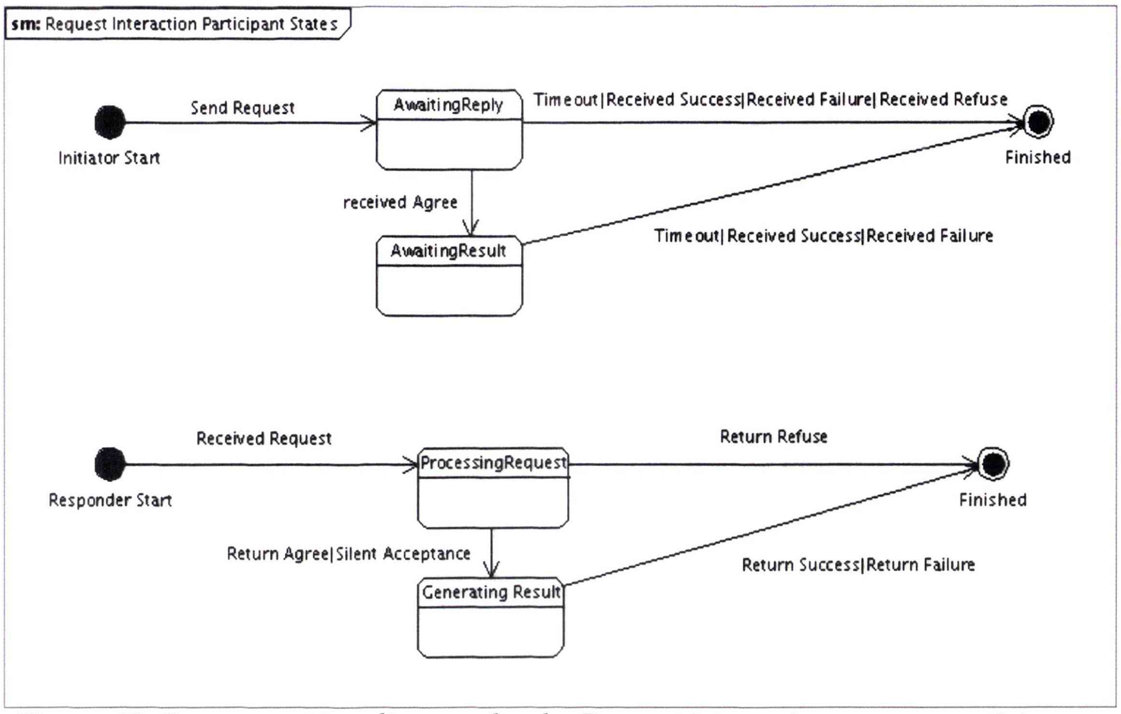

sm: Request Interaction Participant States
Send Request
Initiator Start
AwaitingReply
Timeout|Received Success|Received Failure|Received Refuse
Finished
received Agree
AwaitingResult
Timeout|Received Success|Received Failure
Received Request
Responder Start
ProcessingRequest
Return Refuse
Finished
Return Agree|Silent Acceptance
Generating Result
Return Success|Return Failure

Figure 40: State-transition diagram for the Request interaction pattern participants

5.5.1.2 Inform Interaction Pattern

The Inform interaction pattern is the most simple of the interactions. It consists of one sent message containing a proposition. The key content of the Inform interaction pattern is the `Proposition` sent to the receiver.

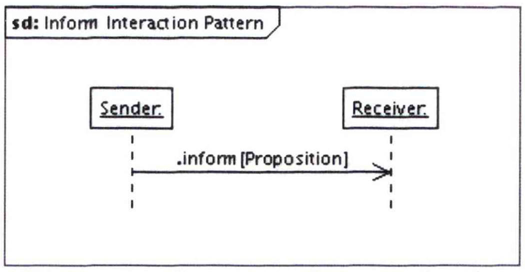


Figure 41: Sequence diagram for the Inform interaction pattern

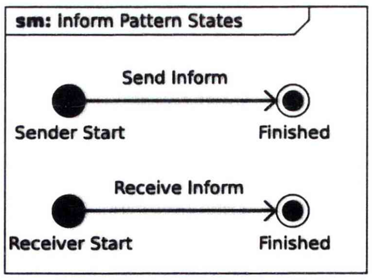


Figure 42: State-transition diagram for the Inform interaction pattern

5.5.1.3 Acknowledged Inform Pattern

The acknowledged inform interaction pattern allows the initiator to send the respondent some information and receive an acknowledgement that the information was received and understood. The key message component in this interaction is the sent `Proposition`.

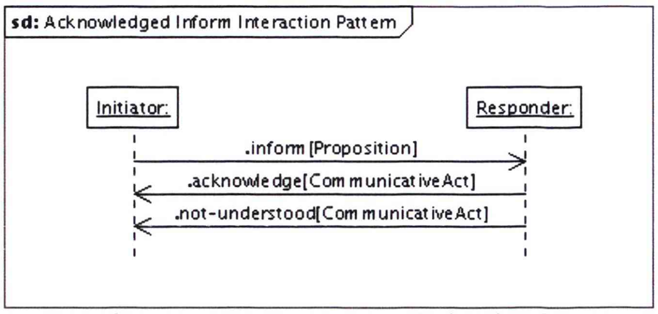

sd: Acknowledged Inform Interaction Pattern
Initiator:
Responder:
.inform [Proposition]
.acknowledge[CommunicativeAct]
.not-understood[CommunicativeAct]

Figure 43: Sequence diagram for the Acknowledged Inform interaction pattern

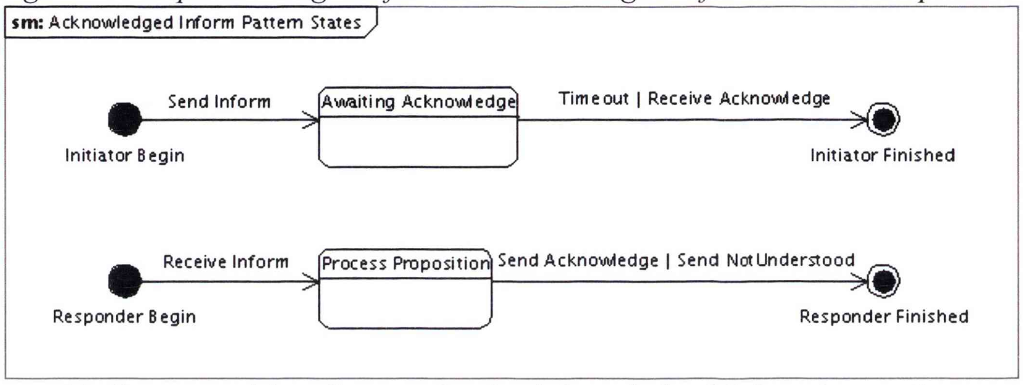


Figure 44: State-transition diagram for the Acknowledged Inform interaction pattern

5.5.2 Agent Dialogues

Each of the standard QDINE dialogues is described below, indicating the interaction pattern for the dialogue and other dialogue specific information. The Key Content of a dialogue corresponds to the key content described in the associated interaction pattern. The full OWL specification of these dialogues can be found in appendix A.2.

5.5.2.1 Consumer Registration

Interaction Pattern:	Request
Participant Actors:	Consumer and Market Agent
Initiator:	Consumer
Key Content:	Requested Action: `RegisterConsumer` (§B.1.3) Success Proposition: `ConsumerRegistered` (§B.2.7.2)
Purpose:	If a negotiation mechanism for a service involves the market agent, the market agent may require consumers to be registered in the market agent before they can participate in such a mechanism.

5.5.2.1.1 Consumer De-registration

Interaction Pattern:	Request
Participant Actors:	Consumer and Market Agent

Initiator: Consumer

Key Content: Requested Action: `DeregisterAgent` (§B.1.6)
Success Proposition: `ActionDone` (§B.2.7)

Purpose: A consumer may wish to de-register itself from a market. On a successful de-registration, the market agent is required to send the consumer a message indicating the successful de-registration.

5.5.2.1.2 Consumer De-registered

Interaction Pattern: Inform

Participant Actors: Consumer and Market Agent

Sender: Market Agent

Key Content: The `ActionDone` (§B.2.7) proposition specifying the `DeregisterAgent` action.

Purpose: A market agent may cancel a registration at any time and is required to inform the consumer when doing so.

5.5.2.2 Provider Registration

Interaction Pattern: Request

Participant Actors: Service Provider and Market Agent.

Initiator: Service Provider

Key Content: Requested Action: `RegisterProvider` (§B.1.4)
Success Proposition: `ProviderRegistered` (§B.2.7.1)

Purpose: Before allowing an agent to publish services within the market or participate in a negotiation mechanism hosted by the market, a market agent may require an agent to register as a provider.

5.5.2.2.1 Provider De-registration

Interaction Pattern: Request

Participant Actors: Service Provider and Market Agent

Initiator: Service Provider

Key Content: Requested Action: `DeregisterAgent` (§B.1.6)
Success Proposition: The `ActionDone` (§B.2.7) proposition specifying the `DeregisterAgent` action.

Purpose:	The Provider De-registration protocol allows a service provider to de-register itself as a provider in a market.

5.5.2.2.2 *Provider De-registered*

Interaction Pattern:	Inform
Participant Actors:	Service Provider and Market Agent
Sender:	Market Agent
Key Content:	The `ActionDone` (§B.2.7) proposition specifying the `DeregisterAgent` action.
Purpose:	At any time a market agent may de-register a service provider from the market and is required to inform the service provider that it has been de-registered.

5.5.2.3 Service Publication

Interaction Pattern:	Request
Participant Actors:	Service Provider and Market Agent
Initiator:	Service Provider
Key Content:	Requested Action: `UpdatePublishedServices` (§B.1.5) Success Proposition: The `ActionDone` (§B.2.7) proposition specifying the `UpdatePublishedServices` action.
Purpose:	The Service Publicatio dialogue allows a service provider to publish services to the market.
Discussion:	Services are published as service specification templates that fully specify an available service. Service obligations(§4.4.3.2) included on created SLAs should include service specification individuals that are of the same class as the published service templates, and should include any property values that are defined on the service template. Conformance to the above guidelines is not enforced by the QDINE framework, but may be required by a service provider. As part of the service publication, the service provider includes the negotiation protocols supported for distributing each published service. A service publication may specify newly available services, services no longer available, and any changes

to currently published services.

5.5.2.4 Agent Authentication

Interaction Pattern:	Request
Participant Actors:	Billing Provider and Authenticating Agent
Initiator:	Authenticating Agent
Key Content:	Requested Action: `AuthenticateAgent` (§B.1.2) Success Proposition: `AgentAuthenticated` (§B.2.7.4)
Purpose:	The Agent Authentication dialogue enables the authenticating agent to authenticate another agent's proxy authentication credentials. The need for proxy authentication is discussed in section 5.2.4.1.

5.5.2.4.1 Agent Authentication Revoked

Interaction Pattern:	Inform
Participant Actors:	Billing Provider and Authenticating Agent
Sender:	Billing Provider
Key Content:	`AuthenticationRevoked` (§B.2.1)
Purpose:	A digital certificate that is supplied by a billing provider, and used to identify an agent may become invalid. One possible reason is if the security of the associated private key is compromised. A billing provider may choose to inform the agents to which it has supplied a certificate that the certificate is no longer valid.

5.5.2.5 Subscriber Registration

Interaction Pattern:	Request
Participant Actors:	Subscriber and Market Agent
Initiator:	Subscriber
Key Content:	Requested Action: `ProvideServiceUpdates` (§B.1.7) Success Proposition: `SubscriberRegistered` (§B.2.7.5)
Purpose:	Within QDINE, services available from a market are only advertised to agents registered as a subscriber with the market agent. Consequently, to discover the services available from a

	market, an agent may wish to register as a subscriber with the market agent.
Discussion:	A market agent may, or may not require an agent to be authenticated to receive service advertisements from the market. If the market agent requires subscriber authentication, and authentication details are not included on a subscriber registration request, the market agent may send a refuse message, stating a reason of `AuthenticationDetailsRequired` (§B.2.8.7).

5.5.2.5.1 *Subscription Update*

Interaction Pattern:	Inform
Participant Actors:	Subscriber and Market Agent
Sender:	Market agent
Key Content:	The statement of `ServicesChanges` (§B.2.4)
Purpose:	Whenever the services available from a market change, the Market Agent may choose to inform the registered subscribers of the service changes.

5.5.2.5.2 *Cancel Subscription*

Interaction Pattern:	Request
Participant Actors:	Subscriber and Market Agent
Initiator:	Subscriber
Key Content:	Requested Action: `DeregisterAgent` (§B.1.6) Success Proposition: The `ActionDone` (§B.2.7) proposition specifying the `DeregisterAgent` action.
Purpose:	A subscriber in a market should be able to cancel its subscription to the service updates and remove itself from the market.

5.5.2.5.3 *Subscription Cancelled*

Interaction Pattern:	Inform
Participant Actors:	Subscriber and Market Agent
Sender:	Market agent
Key Content:	The `ActionDone` (§B.2.7) proposition specifying the

`DeregisterAgent` action.

Purpose: A market agent may cancel a subscriber's registration at any time and must attempt to inform the subscriber when doing so.

5.5.2.6 Consumer Authorisation

Interaction Pattern: Request

Participant Actors: Authorisation Initiator and Authorisation Responder

Initiator: Authorisation Initiator

Key Content: Requested Action: `AuthoriseConsumer` (§B.1.8)
Success Proposition: `ConsumerAuthorised` (§B.2.7.6)

Purpose: The consumer authorisation protocol provides the mechanism by which a consumer may be authorised to participate in service negotiation with market agents and service providers.

Discussion: This dialogue may be initiated by a consumer to a market agent or from the consumer direct to the service provider, as well as from the market agent to a service provider agent on behalf of one or more consumers. When this dialogue occurs between a market agent and service provider, the final statement of consumer authorisation sent by the provider is important and should be digitally signed. This is important because resources may be committed by the market for a consumer, and if the provider can argue that it has not authorised the consumer, any negotiation performed to allocate the resource may need to be repeated at the expense of the participating consumers, providers and hosting market agent.

5.5.2.6.1 Consumer Authorisation Revoked

Interaction Pattern: Acknowledged Inform

Participant Actors: Authorisation Initiator and Authorisation Responder

Initiator: Authorisation Responder

Key Content: `ConsumerAuthRevoked` (§B.2.2)

Purpose: At any time, a service provider may inform a market server that a consumer's authorisation has been revoked, or a service provider may inform the consumer directly. Additionally, a market server may inform a consumer that its authorisation has been revoked in that market.

An acknowledged inform pattern is used because if the interaction occurs between a service provider and market agent, the service provider will desire an acknowledgement from the market for the revoked consumers. This is to remove the service provider's responsibility to the market for those consumers as described in the consumer authorisation protocol (§5.5.2.6).

5.5.2.7 Billing Authorisation

Interaction Pattern:	Request
Participant Actors:	Service Provider and Billing Provider
Dialogue initiator:	Service Provider
Key Content:	Requested Action: `AuthoriseBilling` (§B.1.9) Success Proposition: `BillingAuthorised` (§B.2.7.3)

Purpose: When a consumer requests authorisation to participate in service negotiation, it indicates the entity responsible for its billing. Before a service provider will provide services to a consumer, it may want assurance from the indicated billing entity that it will take responsibility for the specified consumer. The Billing Authorisation Protocol allows a provider to receive such an assurance from a billing provider.

Security: The outcome of the Billing Authorisation Protocol is a statement from the billing provider to the service provider that it will accept the financial responsibility, and handle billing for a (set of) consumers. Relying on this statement, the service provider can provide services to the listed consumers knowing the billing provider will accept the accrued charges. If the billing provider can repudiate a statement of billing acceptance, the service provider may have trouble recovering the charges for services provided to those users. For this reason, the billing acceptance statement should be non-repudiable.

An aspect of non-rupudiability is addressed by the ontology hash ontology (§4.4.2). Instantiating a billing authorisation statement as a `NonRepudiableThing`(§4.4.2.1), and attaching a digital signature to the billing statement message ensures the statement can not be repudiated by the billing provider.

A billing authorisation request includes the agent id to which any revocations for provided authorisations can be sent. The service provider is required to remain contactable via this agent id as long as it provides services for the billing authorisations using that id.

5.5.2.7.1 Billing Authorisation Revoked

Interaction Pattern:	Acknowledged Inform
Participant Actors:	Billing Provider and Service Provider
Initiator:	Billing provider
Key Content:	`BillingAuthRevoked` (§B.2.3)
Purpose:	Whenever a billing provider wants to withdraw its role as the billing provider for a consumer whilst a billing authorisation is active for that consumer, it must send the owners of that billing authorisation a revocation of the billing authorisation for that consumer.
Security:	The outcome of this dialogue is a revocation acknowledgement by the service provider. The revocation acknowledgement must be non-repudiable, as it relieves the billing provider of its financial responsibility for the stated consumer.

As stated in the norms of the service provider role(§5.6.2), a service provider must allow the revocation of active billing authorisations granted by billing providers.

If the service provider does not accept a revocation request, or is not contactable via the revocation address, the billing provider may still choose to dishonour any charges accrued after the billing revocation was sent. In such a situation, when charges continue to be accrued, the service provider will need to take legal action with the billing provider to recover any accrued charges. It is therefore in the service provider's best interest to accept a valid billing revocations sent by a billing provider.

5.5.2.8 Billing Capability Exchange

Interaction Pattern:	Inform or as per Figure 45 (pg.142)
Participant Actors:	Service Provider and Billing Provider

Initiator: Billing Capability Initiator

Key Content: Inform Proposition: `BillingCapabilityStatement` (§B.2.5)

Requested Action: `Inform(BillingCapabilityStatement)`

Purpose: The exchange of billing capabilities is performed to establish the billing methods available between two parties. Either the service provider or billing provider can initiate the exchange, and as such the two actors are simply called BC Initiator and BC Responder. This exchange enables service providers and billing providers to share information as to the billing methods supported, but does not aim to describe the actual billing methods used.

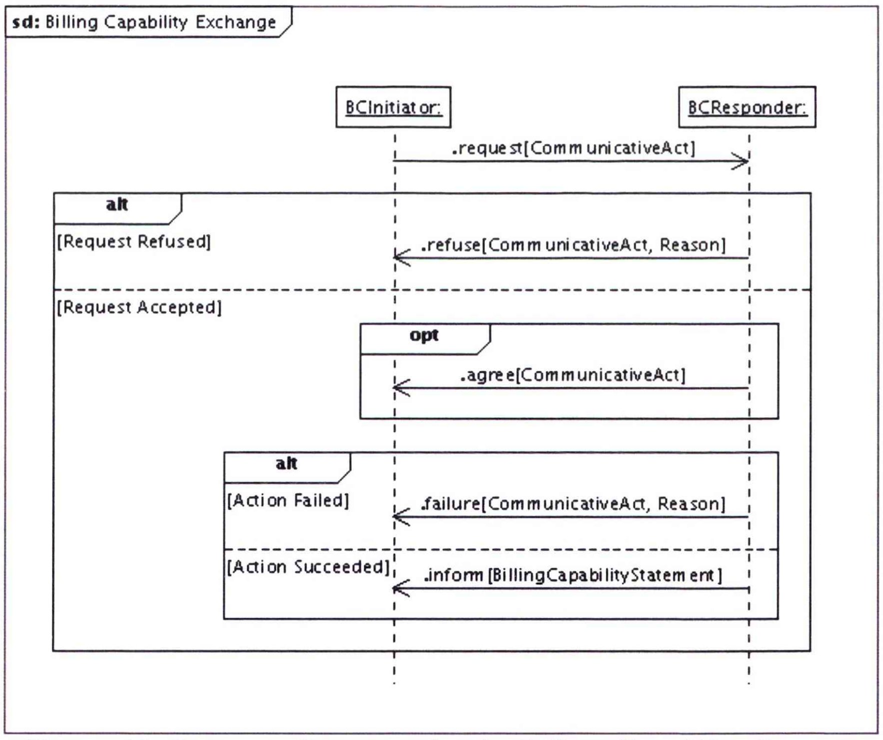

sd: Billing Capability Exchange
BCInitiator:
BCResponder:
.request[CommunicativeAct]
alt
[Request Refused]
.refuse[CommunicativeAct, Reason]
[Request Accepted]
opt
.agree[CommunicativeAct]
alt
[Action Failed]
.failure[CommunicativeAct, Reason]
[Action Succeeded]
.inform [BillingCapabilityStatement]

Figure 45: Sequence Diagram for the Billing Capability Request

Interaction Pattern: Request

Participant Actors: Key Challenger and Key Holder

Initiator: Key Challenger

Key Content: Requested Action: `AnswerKeyChallenge` (§B.1.10)

Success Proposition: `KeyChallengeAnswered` (§B.2.7.7)

Purpose: Authentication within QDINE is based around the Public Key Infrastructure (PKI). As such, the identity of an individual is based on a digital certificate. A certificate may be signed by a trusted certification authority and as such be self supporting, or an agent's certificate may be signed and managed by its nominated billing provider. Both of these mechanisms require some way to ensure the identified agent holds the private key corresponding to the distributed public key on the certificate. Using asymmetric encryption techniques, the key validation dialogue allows an agent acting in a Key Challenger role to verify that another agent (the Key Holder agent role) has the private key corresponding to its distributed public key.

5.5.2.10 SLA Binding

Interaction Pattern:	As per Figure 35 (§5.4.4.1)
Participant Actors:	Consumer and Service Provider
Initiator:	Service Provider
Key Content:	Proposed Action: `ProvideService` (§B.1.11)
Purpose:	The binding process ensures the provider and consumer have a non-repudiable commitment to a negotiated SLA. It is performed after some negotiation process used to create the desired SLA.

5.5.2.11 Negotiation Mechanisms

The appropriate negotiation mechanism for establishing an SLA is dependant on the type of service being negotiated and the context of the negotiation. A finite range of negotiation mechanisms is therefore not appropriate for distributing an unrestricted range of services. A discussion of some example negotiation mechanisms is given to illustrate some SLA distribution models possible with the QDINE framework.

5.5.2.11.1 Two-Stage Request

A two-stage negotiation protocol has been defined for use within the QDINE project. It is similar to the mechanism used by RNAP[142] to allocate network resources. It includes two request/response sequences:

1. The first stage of the protocol consists of a request for proposal, and the reply of one or more offers. In this stage, the Market Agent (MA) performs two functions as the facilitator in the negotiation:
   i. On the outgoing request, the requested service need not be fully specified, and

may have a set of constraints placed on the outcome service agreement. On receipt of the outgoing request, the MA distributes the request to only those providers that satisfy the request constraints.

ii. Each provider then creates complete service offers according to the specified constraints. On receipt of offers from providers, the MA performs an offer selection according to a selection mechanism that was supplied by the user in the original outgoing request.

2. The second stage of the mechanism consists of an offer acceptance sent by a consumer, followed by a signed service level agreement by the provider as a contract to deliver the specified service. In this second stage of negotiation, the interactions occur between the consumer and provider, and the MA is not involved.

5.5.2.11.2 Posted Price Purchase

Using a posted price mechanism, services are fully specified at the time of advertisement, including their charging mechanism and price. Using such a market mechanism, a user sends a purchase request to a provider, potentially via the Market Agent. The provider can return either a success or failure message reflecting the outcome of the purchase. Service advertisements are not binding, but purchase requests are. If a provider advertises services which are not honoured, the Market Agent may choose to deny the provider from future participation, or the user's reputation of the provider may be impacted.

5.5.2.11.3 Bilateral Bargaining

The bargaining approach is suited to 1-on-1 negotiations with multi-issue items. The bilateral bargaining approach is used by SrNP in the TEQUILA[131] and MESCAL[196] projects. With bilateral bargaining, a consumer sends a service request to a service provider, potentially via the Market Agent. The provider then sends a service offer to the consumer, who can either accept the offer or propose a counter offer to the provider. The provider may either acknowledges the consumer acceptance, accept the counter offer or propose another counter offer. Offers sent using such an approach must be fully specified service agreements. The bargaining cycle continues and at any point, a participant can specify an offer as its final offer. Negotiation completes when an involved party either accepts or refuses an offer. Offers and acceptance of offers in a bilateral bargaining are typically binding.

5.5.2.11.4 Combinatorial Clearing House

Using a combinatorial clearing house mechanism, a continuous auction with posted clearing times is conducted by the Market Agent. Providers submit their available resources, and consumers provide their required resources as a set of bids. Every time the market clears, the market mechanism is run generating a set of resource allocations that are sent to the participating providers and consumers. Consumer bids and provider resource availabilities are binding based on this mechanism. This mechanism shares similarities with the Merkato[8] approach to resource allocation for telecommunication networks.

5.6 QDINE Market Roles

Agents participate in QDINE markets by adopting one or more market roles. A *role* is a set of behaviours, rights and obligations [191] as conceptualised by an actor in an interactive environment. There are 5 roles in a QDINE service market:

- Consumer,
- Service Provider,
- Market Agent,
- Subscriber, and
- Billing Provider

Roles are adopted by agents, and any agent may adopt one or more roles provided it maintains the norms of the role. In addition to norms, a role defines a set of expected capabilities of an agent when acting in that role, and a set of rights that it may grant to other roles in the market. A capability is some action that an agent acting in the role may perform.

Within QDINE, agent roles are self assigned. An agent can indicate its adoption of a role to one or more other agents by sending one of the "Indication Messages" defined for that role. An "Indication Message" informs all recipients that the sender intends to adopt the associated role. Self assigned roles allows a distributed, open approach to role assignment.

The use of self assigned roles relies on a solid mechanism for determining the trust of another agent within the market. For example, an agent adopting the role of billing provider, but with little or no reputation or trust in the market may not be accepted as reliable by other players. As stated previously in section 5.3.3, the mechanism by which the trust in another agent is established is not defined by this work. Electronic

Institutions[190] offer a mechanism by which the appropriate level of trust for a participating agent acting in some role can be ensured by the institution.

The use of self assigned roles does not preclude the market agent from maintaining a registry of agent ids and associated roles. The market agent may desire such a registry to monitor the size of its market for marketing purposes.

Each of the role norms, indication messages, capabilities and grantable rights are outlined below, specifying the appropriate QDINE dialogues where applicable:

5.6.1 Consumer

Consumers enter a market with the intention of purchasing SLAs from one or more service providers.

Indication Messages

The following messages indicate the sender's intention to adopt the Consumer role:

- The `RegistrationRequestMsg` (§A.2) within the Consumer Registration protocol. (§5.5.2.1)
- The `ConsumerAuthRequestMsg` (§A.2) within the Consumer Authorisation protocol (§5.5.2.6), when the authentication credentials on the registration request are the same as those in the consumer details on the registration request.

Capabilities

Consumers in a QDINE market may choose to perform any or all of the following actions:

- Register with a service market ("Consumer" Actor, §5.5.2.1).
- De-register from the service market (Consumer, §5.5.2.1.1).
- Request agent authentication from a billing provider (Authenticating Agent, §5.5.2.4)
- Authorise itself as a consumer with a service providers or the market agent (Authorisation Initiator, §5.5.2.6).
- Validate an opponents public key (Key Challenger, §5.5.2.9).
- Participate in negotiations for service level agreements.
- Bind Service Level Agreements (Consumer, §5.5.2.10)

Norms

An agent acting in the Consumer role is required to satisfy the following:

- Enable validation of a distributed public key by supporting the key validation protocol (Key Holder, §5.5.2.9).

Grantable Rights

None

5.6.2 Service Provider

The Service Provider market role publishes its available services to the market with the intention of providing SLAs to consumers for those services. Any agent may adopt the service provider role. That is, adoption of the service provider role is not restricted, and may be as simple as an entity wishing to share their wireless Internet connection.

Indication Messages

The following messages indicate the sender's intention to adopt the Service Provider role:

- The `RegistrationRequestMsg` (§A.2) within the Provider Registration protocol. (§5.5.2.2)
- The `BillingAuthRequestMsg` (§A.2) within the Billing Authorisation protocol (§5.5.2.7)
- The `ConsumerAuthAgreeMsg`, `ConsumerAuthRefuseMsg`, `Consumer-AuthFailureMsg` or `ConsumerAuthSuccessMsg` (§A.2) within the Consumer Authorisation Protocol (§5.5.2.6).

Capabilities

Service Providers in a QDINE market may choose to perform any or all of the following actions:

- Register with a service market (Service Provider, §5.5.2.2).
- De-register from the service market (Service Provider, §5.5.2.2.1).
- Maintain services in the service market (Service Provider, §5.5.2.3).
- Request agent authentication from a billing provider (Authenticating Agent, §5.5.2.4)
- Authorise consumers to purchase services (Service Provider, §5.5.2.6)
- Revoke consumer authorisations (Authorisation Responder, §5.5.2.6.1).
- Request billing authorisation from a billing provider (Service Provider, §5.5.2.7).
- Exchange billing capabilities with a billing provider (BC Initiator or Responder, §5.5.2.8).
- Participate in negotiations for service level agreements.
- Bind Service Level Agreements (Provider, §5.5.2.10)

Norms

An agent acting in the Service Provider role is required to satisfy the following:

- If registered with a market agent, then have at least 1 service published in the market.
- Ensure information published about services is correct. This involves:
  - ✧ Removal of publications for services that are no longer available.
  - ✧ Modification of publications for services that have changed.
- For any current billing authorisations, accept billing authorisation revocations from billing providers. This includes:
  - ✧ Supporting the billing authorisation revocation protocol (Service Provider, §5.5.2.7.1).
  - ✧ Maintaining reachability via the revocation address listed on the original billing authorisation request.
- Support the Consumer Authorisation protocol as the Authorisation Responder actor (§5.5.2.6).
- Support the key validation protocol as the "Key Holder" actor. (§5.5.2.9)
- Participate in negotiations for service level agreements in accordance with the negotiation mechanisms attached to published services.
- If a consumer's authorisation is revoked, inform all agents who have previously been sent an authorisation for that consumer. (Authorisation Responder, §5.5.2.6.1).

Grantable Rights

A Service Provider may grant the following rights to other agents in a QDINE market:

- Authorise an agent as a consumer for provided services.
- Allow a consumer to negotiate for a provided service.
- Provide billing capabilities to a billing provider.

5.6.3 *Market Agent*

The market agent role is performed by an agent hosting a market for services. The market agent role defines a service market. As such, there can be only one market agent per service market. If an agent decides to play the role of a service market, a new service market is created around that agent.

Indication Messages

The following messages indicate the sender's intention to adopt the Market Agent role:

- The `RegistrationAgreementMsg`, `RegistrationRefusedMsg`, `RegistrationFailureMsg` or `RegistrationSuccessMsg` (§A.2, A.2, A.2) within the Subscriber, Consumer or Provider Registration Protocols (§5.5.2.5, 5.5.2.1 & 5.5.2.2).

Capabilities

The Market Agent of a QDINE market may choose to perform any or all of the following actions:

- Register subscribers to the market (Market Agent, §5.5.2.5).
- Advertise available services to subscribers (Market Agent, §5.5.2.5.1).
- De-register subscribers from the market (Market Agent, §5.5.2.5.2 & §5.5.2.5.3).
- Register consumers to the market (Market Agent, §5.5.2.1).
- De-register consumers from the market (Market Agent, §5.5.2.1.1 & §5.5.2.1.2).
- Request authorisation for consumers to receive services from service providers (Authorisation Initiator, §5.5.2.6).
- Authorise consumers to participate in the market (Authorisation Responder, §5.5.2.6).
- Revoke consumer authorisation for an agent in the market (Authorisation Responder, §5.5.2.6.1).
- Request agent authentication from a billing provider (Authenticating Agent, §5.5.2.4)
- Register service providers to the market (Market Agent, §5.5.2.2).
- De-register service providers from the market (Market Agent, §5.5.2.2.2 & §5.5.2.2.1).
- Collect available services from providers (Market Agent, §5.5.2.3).
- Participate in, and possibly host negotiation mechanisms for service level agreements.
- Act as a proxy for interactions between local consumers and agents from neighbouring markets (§5.4.5.2).

Norms

An agent acting in the Market Agent role is required to satisfy the following:

- Support the consumer registration protocol (§5.5.2.1), including consumer de-registration (§5.5.2.1.1)
- Inform consumers if they have been involuntarily de-registered (§5.5.2.1.2).
- Support the provider registration protocol (§5.5.2.2), including provider de-registration (§5.5.2.2.1)
- Inform service providers if they have been involuntarily de-registered (§5.5.2.2.2).
- Support the subscriber registration protocol (§5.5.2.5), including the cancel subscription dialogue (§5.5.2.5.2).
- Inform subscribers if their subscription has been involuntarily cancelled (§5.5.2.5.3).
- Support the service publication protocol (§5.5.2.3).

- Allow service providers to revoke a consumer authorisation previously granted to the Market Agent. That is, support the consumer authorisation revocation protocol (§5.5.2.6.1) as the "authorisation initiator" actor.
- If a consumer's authorisation in the market is revoked, inform the consumer of the revocation. (Authorisation Responder, §5.5.2.6.1). Note: This does not mean that the consumer should be notified whenever a single provider revokes the consumer's authorisation, only when the market revokes the consumer's authorisation to participate in the market.
- Ensure the information that subscribers have about advertised services is accurate. A market agent is required to:
  - ✧ Withdraw service availability for previously advertised services which have since become unavailable.
  - ✧ Propagate any changes to service descriptions for previously advertised services.
  - ✧ Ensure all pre-negotiable services advertised have the `originatingMA` property populated with a valid agent ID. For services published by local providers with an empty `originatingMA` property, the market agent may use its own agent ID. (§5.4.5.2)
- Support the key validation protocol as the "Key Holder" actor. (§5.5.2.9)

Grantable Rights

A Market Agent may grant the following rights to other agents in a QDINE market:

- Register subscribers in the market.
- Register consumers in the market.
- Authorise consumers to participate in the market.
- Register service providers in the market.
- Allow service providers to publish services.
- Advertise services that are published by providers.
- Host negotiations for services.
- Allow agents to participate in hosted negotiation mechanisms for service level agreements.
- Act as a proxy for local consumers to interact with agents from neighbouring markets (§5.4.5.2).

5.6.4 Subscriber

A subscriber role is adopted by any agent that wants to be informed of the services available from a market agent.

Indication Messages

The following messages indicate the sender's intention to adopt the Subscriber role:

- The `RegistrationRequestMsg` (§A.2) within the Subscriber Registration protocol (§5.5.2.5).

Capabilities

Subscribers in a QDINE market may choose to perform any or all of the following actions:

- Subscribe to service update notifications (Subscriber, §5.5.2.5).
- Cancel service update subscriptions (Subscriber, §5.5.2.5.2).
- Request agent authentication from a billing provider (Authenticating Agent, §5.5.2.4)

Norms

An agent acting in the Subscriber role is required to satisfy the following:

- Support the key validation protocol as the "Key Holder" actor. (§5.5.2.9)

Grantable Rights

- None

5.6.5 Billing Provider

The billing provider role has two key responsibilities: Authenticating a consumer's identity who doesn't have a trusted digital certificate, and providing a reliable point of billing for service providers.

Indication Messages

The following messages indicate the sender's intention to adopt the Billing Provider role:

- The `BillingAuthAgreeMsg`, `BillingAuthRefuseMsg`, `Billing-AuthSuccessMsg` or `BillingAuthFailureMsg` (§A.2) sent from the Billing Authorisation Protocol (§5.5.2.7)
- The `AgentAuthAgreeMsg`, `AgentAuthRefuseMsg`, `AgentAuth-SuccessMsg` or `AgentAuthFailureMsg` (§A.2) sent from the Agent Authentication Protocol (§5.5.2.4)

Capabilities

Billing Providers in a QDINE market may choose to perform any or all of the following actions:

- Authenticate an agent by generating and distributing a signed digital certificate

(Billing Provider, §5.5.2.4).
- Inform agents that an authentication certificate has been revoked (Billing Provider, §5.5.2.4.1).
- Commit itself to a service provider as the responsible billing entity for a consumer (Billing Provider, §5.5.2.7).
- Revoke billing authorisations (Billing Provider, §5.5.2.7.1)
- Exchange billing capabilities with a service provider (BC Initiator or Responder, §5.5.2.8).

Norms

An agent acting in the Billing Provider role is required to satisfy the following:

- Support the Agent Authentication protocol (Billing Provider, §5.5.2.4).
- Support the Billing Authorisation protocol (Billing Provider, §5.5.2.7).
- Revoke a billing authorisation certificate whenever the associated private key is compromised (Billing Provider, §5.5.2.7.1).

Grantable Rights

A Billing Provider may grant the following rights to other agents in a QDINE market:

- Commitment as the responsible billing entity for a consumer.
- Provide billing capabilities to a service provider.
- Authentication of contracted agents' credentials.

5.7 Operational Market Concerns

5.7.1 Knowledge Consistency

Messages passed between agents are an exchange of information. Within QDINE, information is expressed formally in a knowledge representation language. If adopting the contents of a received message will cause the receiver's knowledge base to become inconsistent, the receiver must choose how to handle the potential inconsistency. An agent may simply override the previous knowledge, ignore the new knowledge or use some other hybrid strategy. Following the concept of agent autonomy, no single agent action is prescribed for such a scenario and any action may be taken, however the following two actions are directly supported by the QDINE framework.

If the receiving agent does not adopt the new conflicting knowledge, the receiving agent may reply to the sender, using the "not understood" message performative, citing a reason of `ConsistenceViolation`(§B.2.8.15). The new knowledge causing the inconsistency may optionally be included, along with the existing knowledge which is in conflict. An agent is not required, and may choose to not share either the new or existing knowledge in conflict, as it may be confidential or offer some unwanted advantage to the sender.

If the knowledge conflict is a result of a newly defined individual on the sent message with the same RDF ID as an existing individual, and the message receiver chooses to override the existing individual with the new individual, then the receiver may send an "inform" message to the sender, including an `OverriddenKnowledge` statement(§B.2.6), listing the OWL Individuals that were overridden.

A further consideration in the area of knowledge consistency is that signed SLAs should not refer to any QDINE specific resources (§5.2.1.3.1) that are defined externally to the signed block that contains the SLA definition. This is because the signed content block containing the SLA is all that is guaranteed to be saved after the SLA negotiation process has completed. Any QDINE specific resources that are defined externally to the signed block may be lost as QDINE specific resources are temporary and are not archived by the ontology hash mechanism(§4.3.2).

5.7.2 Stakeholders

The following discussion identifies four key stakeholders within a QDINE market: End Users, Service Providers, Network Operators and Billing Providers. This list of stakeholders is not exhaustive, and other stakeholders may appear in a market, adopting one or more of the QDINE roles to support their business model. Each stakeholder represents itself as one or more agents in the market. On implementation, an agent may be composed of multiple sub agents performing dedicated tasks. The division of tasks to sub agents is not the concern of this work and is not addressed further.

5.7.2.1 End Users

An end user may be a single individual with some form of network access device, or an organisation that uses services for conducting business. End users do not provide services to the market and may play the following roles in a QDINE market:

- Consumer – The primary motivation of an end user joining a market is to act as a consumer, requesting services from providers in the market.
- Subscriber – To become aware of the services available in a market, an end user may act as a subscriber in the market and receive service updates.

5.7.2.2 Service Providers

Within the current telecommunications value chain, service providers are typically limited to ISPs or mobile service providers. Currently, the network service provider acts as a proxy for any supporting services such as video content or weather services. Within a QDINE market, the current network service providers will exist, however, it is expected that further service providers will emerge and participate directly in the service value chain. Some examples are storage providers and application service providers. Service providers may play the following market roles:

- Service Provider – To make its services available in the market.
- Consumer – When a service offered depends on other services available in the market, a service provider will act as a consumer for those services.
- Subscriber – To become aware of the services available in a market, a service provider may act as a subscriber in the market.
- Billing Provider – When acting as a consumer, if the service provider supports direct billing, it may to allow other service providers to bill it directly for services.

5.7.2.3 Network Operators

Network operators are a special type of service provider, in the unique position of controlling the network resources required for other services to operate. The QDINE market roles likely to be performed by a network operator are:

- Service Provider – To make its networking resources available as a service direct to consumers as well as other service providers.
- Market Agent – The network operator is in an optimal position to provide the service market. This is because it has control over the physical resources to which end-users connect, providing anyone connected to the network with access to the market.
- Subscriber – To establish end-to-end services, a network operator may subscribe to service updates from neighbouring markets.
- Consumer – When a service offered depends on other services available in the market, a network operator may act as a consumer for those services. An example is an end-to-end connectivity service where the network operator will require services advertised from a neighbouring domain.
- Billing Provider – If a network operator can support billing for services it has consumed as a Consumer, the network operator may also act in the billing provider role.

5.7.2.4 Billing Providers

Users desire simplicity and predictability [197]. In terms of billing, a roaming end-user will not want to pay individually each domain which has provided (a fraction of) a service or service session. This creates a role for billing aggregators and speculators. Billing aggregators package provided services into one bill, whilst speculators speculate on future costs and so quote a fee for some future period. Say the next six months. Speculation offers a method to manage risk and gain in an economic system.

A player, referred to as a billing provider, may be both an aggregator and a speculator, with end-users appointing such a player to manage billing on their behalf. Every end-user may have an associated Billing Provider whose status as a trusted and reliable player is generally known. The billing provider acts as a financial proxy for its customers to visited service providers. The resources of a billing provider are its reputation, and the provided proxy financial responsibility on behalf of a user. In addition to the functions of aggregation and speculation, the Billing Provider is responsible for authenticating the end-user to foreign networks as required. Within a QDINE market, a billing provider as described above may perform the "Billing Provider" (§5.6.5) QDINE market role.

5.8 Summary

Within this chapter, a framework for Service Level Agreement negotiation has been presented. The framework builds on the concept of a service market in which an agent may participate in one or more market roles.

Within a QDINE market, agents communicate using FIPA-ACL, with message content expressed in OWL-CL. The market processes are described, along with a set of dialogues that are used by agents to carry out the standard market interactions. These dialogues are described using the protocol description language. Agent behaviour is guided by norms defined for the market roles, norms of the interaction protocols and norms defined on the market itself.

Participation in a QDINE service market enables providers to publish their services, and consumers to negotiate for services using mechanisms that are appropriate to the context of the available services. Along with managing distribution of SLAs, the framework supports secure identification of players and the establishment of billing relationships in an ad-hoc fashion. The characteristics of this framework are now examined in chapter 6, along with comparisons to the current art.

INTENTIONALLY BLANK

Chapter 6

Validating the QDINE Approach

Due to the size, scope and nature of the problem addressed here, complete validation of the QDINE work can not be performed in the conventional scientific sense. That is, it is infeasible to build a complete QDINE-based management system on the telecommunications network, and to observe its performance. In due course, as this work was performed in a collaborative project with an industry partner, the validation will be achieved through adoption of QDINE in a production environment.

The QDINE work is validated in the following three senses:

- First, in section 6.1 we show that QDINE satisfies the requirements for an appropriate SLA specification and negotiation framework as stated and justified in chapter 3.
- In section 6.2 we argue that the process by which the QDINE solution has been derived, in particular the set of decisions made in assembling the QDINE machinery took full account of the relevant state of the art in IT.
- Third, in section 6.3 we argue that QDINE is superior to the existing art in the area of generalised SLA management.

To help make any underlying assumptions explicit, an experimental system was developed and is described in section 6.4. Further validation of the SLA ontology follows in section 6.5 where the generic set of ontologies are extended to create a Service Level Agreement use case for telecommunication services.

6.1 Achieving the Aims

As stated in chapter 3, the two key aims of the QDINE work are to:

- Create a secure, flexible, semantic Service Level Agreement description language, and
- Create a secure, open, flexible market-based service level agreement negotiation framework

A detailed set of requirements for meeting the above aims were proposed in chapter 3. The way in which the QDINE work addresses each of the requirements is described below.

6.1.1 A Secure, Flexible Semantic SLA Description Language

This aim has been achieved by the SLA Ontology. The SLA ontology enables the creation of SLAs using OWL; a *machine intelligible* knowledge representation language with *formal semantics*. OWL is a W3C *standard* language and the SLA ontology will be released under a Creative Commons Attribution 2.5[23] *open* copyright license, allowing loyalty-free re-use and extension by other parties.

The SLA ontology is divided into multiple, small and decoupled definition files, with only highly related concepts grouped together. Any service-specific concepts are confined to the "service specification" in an agreement, and any type of service specification may be used in an agreement. Further, OWL provides inheritance, allowing generic concepts to be *extended* into a particular application domain. The ontology is therefore *highly flexible* and may be used to describe agreements for any type of service.

The SLA ontology is *independent* of a particular delivery or negotiation framework. The SLA ontology describes the layout and semantics of an SLA, but the way in which an SLA is created and distributed is decided by the adopters of the ontology. Additionally, the generic structure of the SLA ontology and the inclusion of core concepts such as obligation, entities and schedules, provides a solid foundation for re-use, while allowing domain specific extensions as desired. This allows an SLA to *stand alone*, including all information required to operate and administer a described service, with no reliance on externally described information.

Through combination of the described Ontology Hash mechanism and digital signatures for service level agreements, the SLA ontology can be used to create *secure* agreements. The agreements have an explicit and verifiable acceptance on the semantics of the agreement at the time of signing, providing strong non-repudiation.

Through the use of the Unit and Metric ontologies, the parameters in a service level agreement are *grounded*, and can be associated with real-world measurement techniques and compared with specified targets if desired. Additionally, the set of SLA ontologies has a well defined relationship to an Event Calculus[172], allowing automated reasoning over temporal concepts used on an SLA.

Finally, because globally unique ids and semantic relations are integral features of OWL, the information sent within a negotiation sequence is *compact*. The reason for this is that transmitted data need only be a pre-existing OWL individual or class

23 http://creativecommons.org/licenses/by/2.5/

identifier plus any modifications from that individual. For example, if a service is described using OWL, given an identifier, and made available on the Web, then only the URI of that service is required on any SLAs created for that service. The full service description may be retrieved at any time by querying the service URI.

6.1.2 A Secure, Open, Flexible Negotiation Framework

This aim has been addressed by the QDINE negotiation framework. The QDINE framework is founded on a philosophy of agent *autonomy*. All agents are ultimately responsible for there own resources and behaviour, and may participate in the framework with no reliance on other parties. Relationships with other agents are not necessary, but may be established to provide some extra capability, such as increased trust, reputation or simplified billing.

No assumption is made on the internal workings of an agent operating within a QDINE market. An agent is *free* to act as it sees fit, and is guided by the norms of the market. If desired, a service negotiation mechanism may employ the Market Agent as a *supervisor* of messages sent between agents. This can be used to ensure the behaviour of agents complies with protocol norms within a complex or sensitive negotiation scenario.

The QDINE framework provides a *secure* environment in which players can interact. Players can be identify themselves through a trusted digital certificate, or can nominate a billing provider to verify their identity to opponents. The use of an XML-based communication language and the FIPA communication framework allows the use of security techniques applicable to both XML *and* FIPA frameworks. Existing techniques are available which address privacy, integrity and non-repudiation. Additionally, the use of a formal knowledge representation language; namely OWL, allows the validity of the communicated messages to be tested against a common and formally specified set of constraints.

The QDINE framework specifies the use of OWL-CL as a content language, but makes no restrictions as to the required content of messages or the structure of an SLA. The framework is functionally *separate* from the contents of an SLA. As long as a desired SLA can be described using OWL, it can be distributed using the QDINE framework. Additionally, the use of FIPA ACL allows maximal *flexibility* in agent interaction. Any pattern of interaction may be used for service negotiation involving any number of participants. The QDINE framework is independent of charging, billing or pricing systems, but allows participants to establish the relationships required to bootstrap the financial chain required for service delivery.

Using the FIPA communication mechanism and a content language based on OWL, the QDINE framework has adopted widely accepted *standards* for interaction between participants. FIPA-ACL is generally considered the standard agent communication language [154]. The use of these technologies also provides a significant abstraction from the technical implementation used by participants, allowing different implementations to interact over a common communication platform. The technical abstraction of communication by the QDINE framework and logical abstraction of services by the SLA ontology, along with the service advertisement and negotiation support systems available within the QDINE framework support the *ubiquitous* provisioning of services across heterogeneous networks.

Finally, the use of an agent-based system and formal semantics for both the communication mechanisms and the service level agreements provides a sound foundation for automated reasoning over information received by agents. Automated reasoning can support automated decision making, and ultimately automated management of services by intelligent agents.

6.2 The QDINE Solution Derivation Process

In line with the requirements listed in chapter 3, many decisions were made while forming the solution proposed in this work. When existing work was found in a particular solution space, decisions were made whether to adopt the work entirely, adopt some concepts explored in work, or develop a new solution. Whenever appropriate standards were available, these were typically adopted.

6.2.1 The SLA Ontology

After identifying the need for a service level agreement description language with formal semantics, existing work in the area was analysed, and the final set of requirements as listed in chapter 3 was created.

The main languages considered were UML, KIF, XML, RDF, DAML and OWL. OWL was chosen because of its machine readable format, ability to explicitly describe semantic knowledge, integration with web based technologies and the semantic web, and standardisation by the W3C.

The decision to adopt OWL as the SLA description language prompted a survey on the existing OWL ontologies related to Service Level Agreements. No appropriate OWL ontologies were found that could express an SLA or service obligation. Ontologies were found that express supporting concepts such as events and units [198][199][200][201], however many concepts were absent or composed in a way that do not suit the SLA structure. The Suggested Upper Merged Ontology (SUMO)[200] and Mid-Level Ontology[201] are generic ontologies and have some concepts compatible with the requirements of QDINE. In future work, the QDINE concepts may be mapped onto the concepts expressed in SUMO to aid interoperability. This highlights a further advantage of using OWL, that is the ability to semantically map concepts from one ontology to another.

6.2.2 The Negotiation Framework

6.2.2.1 Communication Mechanisms

In much of the research in telecommunication service negotiation, the systems use a fixed communication mechanism for service negotiation and distribution. These fixed mechanisms limit interoperability between platforms. For a service negotiation platform to be ubiquitous, what is required is a platform that can use negotiation mechanisms appropriate to the service context. In a departure from the telecommunication service negotiation literature, and borrowing from theory in generalised negotiation [7], it was decided to adopt a generalised communication approach, relying on a formal specification of the communication protocols and the use of speech acts between agents. This allows the use of different service delivery mechanisms, rather than the "one size fits all" approach typically adopted.

The Foundation for Intelligent Physical Agents (FIPA)[202] created a standard model of communication based on speech act theory and an explicit indication of the protocol in use for a conversation. As a standard, the FIPA model is used by many agent systems, and extensions have been defined such as those for encryption and non-repudiation[185]. The FIPA standard communication model aligned perfectly with the requirements of the QDINE framework and so was adopted.

6.2.2.2 Choosing the Content Language

The FIPA model allows the use of different content languages within messages. Once the decision was made to create a FIPA standard framework, it was necessary to decide which content language would be used, as no FIPA standard content language has native OWL support. FIPA-RDF0[203] is defined for RDF content and since OWL can be expressed in RDF, FIPA-RDF0 was examined as a candidate. FIPA-RDF0 is in an experimental stage and is not yet a standard. Within FIPA-RDF0, no distinction is made between different proposition types such as Statement or Condition, complicating the validation of message content compliance with the constraints imposed by FIPA performatives. A further issue with the FIPA-RDF0 specification is explained in the following: The RDF specification states that the scope of an identified RDF resource with no explicit or default namespace given is limited to the containing document. Within the FIPA-RDF0 specification, no direction is given on what the default namespace for use with documents should be, and no requirement for explicit namespaces is stated, however it is assumed that resources defined in one message (an RDF document) are available within other messages.

The Travel Agent Game in Agentcities (TAGA) has specified an OWL-based content language ontology[204][205]. After analysis of the TAGA content language, the following points were noted:

- The ontology has been used successfully in a trading game.
- The Action class is fully specified, thereby reducing the reuse of the language.
- Acts are encoded as strings, reducing the semantic strength of the content language.
- Action properties are specified as string data types. Datatype properties discontinue semantic continuity and consequently any formal meaning of the property values.

In summary, the TAGA OWL based content language is a useful OWL content language, however it has been built for use in a specific purpose and is overly specific to allow the implementation freedom necessary as a generic content language.

Using both the FIPA-RDF0 and the TAGA OWL content languages as a guide, the decision was made to simply use OWL as the content language as is done with the TAGA OWL content language, however the core ontology created that describes concepts such as actions, conditions and statements is left maximally generic, allowing it to be used within any application domain.

6.2.2.3 Choosing the Protocol Description Language

The first step in specifying a machine understandable set of protocols was to choose a language in which to describe the protocols. Benyoucef and Keller in [206] evaluate five major techniques and formalisms for describing negotiation processes, concluding that the State-chart formalism is most suited to a general set of requirements.

In consideration of automated reasoning, an ontological approach was continued when selecting the Protocol Description Language. An ontological representation of statechart concepts is given in [207] and [208]. However it is poorly documented, does not capture the place of actors within a protocol, and does not consider semantics for the conditions on a transition. Focussing on interaction protocols, two ontology based protocol specification languages were studied:

- An ontology designed for specifying interaction protocols is proposed in [176]. The ontology is represented graphically, and is intended for a DAML implementation. It describes interaction protocols as finite state machines, modelling the transitions as messages, including choices and timeouts. The ontology was never completed or formally described in DAML but presented a strong proof of concept and a clear description of a knowledge-based protocol representation language.
- Desai et al in [177] have developed an ontology named OWL-P to model processes, including a sub ontology for specifying protocols. The ontology uses ruleML [123] for specifying the workings of a protocol. Emphasis is placed on commitments made by agents when receiving or sending a message. The ontology attempts to abstractly model the business logic that participants must adhere to when abiding by the protocol, without actually specifying *how* the agents implement the logic.
  A major drawback with OWL-P is the over-encapsulation of entire steps of a protocol flow into a single SWRL rule. As such, the structure of the protocol is entirely represented by an SWRL rule. This approach provides maximal flexibility, but does little to provide structure by which a protocol can be analysed. Additionally, constraints placed by SWRL are ignored. Two examples of this is the fact that a variable may exist in the consequent of a rule, but not in the antecedent; and the consequent action(s) do not necessarily have to occur whenever the antecedent becomes true. OWL-P also assumes a BDI model of agents which is an unnecessarily specific constraint.

The benefits and limitations of the available ontology-based protocol modelling tools were studied and a new protocol specification language based on OWL was created and named the Protocol Description Language(§5.2.3). This generic protocol description language is not specific to the QDINE project and was designed for modelling protocols in any domain.

6.3 QDINE vs the Existing Art

Previous work in SLA specification and service negotiation platforms has overlooked many of the requirements described in chapter 3, or they have been left for future work. This work extends the current art by addressing all of these requirements, and so makes a significant contribution to automated service management.

Tables 4 and 5 provide a summary of key projects from the current art in consideration of the requirements from chapter 3. A comparison with the QDINE work is also included.

	Machine Intelligible	Formal Semantics	Flexible & Extensible	Independent	Open Standards	Standalone	Secure	Grounded	Compact
SLAng	◒	◒	◒	?	✔	✘	✘	✘	✘
WS-Agreement	✔	✘	✔	◒	✔	◒	✘	✔	✔
SSS	?	✘	✘	✔	?	✘	✘	◒	?
P1008	✔	✘	✘	✘	◒	✘	✘	✘	?
Jin05	✔	✔	✔	◒	✔	◒	✘	✔	✔
SlaML	✔	✘	◒	✔	✔	◒	✘	✘	✘
SLA Ontology	✔	✔	✔	✔	✔	✔	✔	✔	✔

Table 4: The SLA Ontology vs the current art

✘	= Not Supported
◒	= Partial Support
✔	= Full Support
?	= Not Addressed / Could not be ascertained

	Autonomous Responsibility	Security					Separation	Flexibility	Freedom	Open Standards	Ubiquitous	Automated
		Identification	Confidentiality	Integrity	Non-Repudiability	Supervision						
P1008	✘	✔	?	?	◒	✘	✘	✘	✘	✘	✘	◒
Web Services	?	◒	✔	✔	◒	✘	✔	✘	✔	✔	✘	✔
Paurobally05	?	◒	✔	✔	◒	✘	✔	◒	✔	✔	✘	✔
IMS	✘	✔	✔	?	?	✔	✘	✘	✘	✔	◒	◒
Chieng et al.	✔	✘	✘	✘	✘	✔	✘	✘	✘	◒	✘	✔
SLAT	✔	✘	✘	✘	✘	◒	✘	◒	✘	✘	✘	✔
AQUILA	?	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✔
Yan2006	✔	✘	✘	✘	✘	◒	✘	✘	✘	✔	✘	✔
Ouelhadj2005	✔	✘	✘	✘	✘	✘	✘	✘	◒	✘	✘	✔
QDINE	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔

Table 5: The QDINE Framework vs the current art

6.3.1 In comparison with Web Services

The negotiation of services and service level agreements has received substantial attention from the research community. In relation to the stated requirements of this work, the web services framework is the closest approach to meeting the requirements, and is the closest art to the QDINE approach. The QDINE approach is compared directly with the Web Service approach in the paragraphs below.

Support for negotiation

The web services solution to SLA management is the WS-Agreement. A WS-Agreement is a stateful web service resource that is made available via a set of Web Services and specified using the WS-Resource Framework[209]. A WS-Agreement is manipulated by a set of web services that can create, query the current state of, and destroy the agreement. The WS-Agreement specification describes both an SLA agreement schema and the actions that can be performed in relation to agreements. The components of a WS-Agreement are expressed in XML schema, and the operations possible on the SLAs are described in WSDL[210]. The WS-Agreement specification document[143] clearly states that WS-Agreement is not concerned with agreement negotiation.

The first significant weakness of the web services framework is the lack of support for agreement negotiation. Messages for creating a WS-Agreement are limited to two types; offer and agree. Consequently, WS-Agreement service delivery models are limited to those such as buying from catalogues, with take-it or leave-it offers from the seller or buyer. The QDINE communication approach enables rich interaction and the use of any service delivery model, including but not limited to: posted price mechanisms, auctions, exchanges and bilateral bargaining.

SLA specification / management system interdependence

The WS-Agreement specification states that a WS-Agreement can be used to describe agreements for any type of service. This may be possible, however the WS-Resource describing the agreement can only be accessed via the Web Services Framework. As a result, the independence of the SLA specification from the delivery mechanism is compromised. The SLA specification described in this dissertation is deliberately independent from the QDINE negotiation framework, allowing the SLA specification to be used within any desired management system.

Reciprocal commitments

A service level agreement should describe a *pair* of commitments made between a provider and consumer. The WS-Agreement specification does not address reciprocal commitments made by a consumer to a provider, and so must rely on some other external means to express these commitments. The QDINE SLA explicitly includes a pair of commitments for every service obligation included in an SLA.

The significance of a market agent

A novel element of QDINE is the functionality provided by the market agent. The web service framework has a service registry agent, however, such an agent is not aware of, and cannot actively participate in or moderate a service negotiation, as is done by the Market Agent in QDINE.

Comprehensive use of explicit formal semantics

An advantage of QDINE over the Web Services approach is the formal semantics embedded into all aspect of the solution. Services are described using formal semantics, as are the service agreements, the communication language, and the interaction protocols. The advantage of explicit formal semantics becomes evident when automated reasoning is employed. Without semantic annotation, reliable natural understanding is required which as yet is an open research issue in the artificial intelligence community. Additionally, without explicit semantics, the semantics must be assumed to be understood by involved parties or are gathered from some external formal description.

Reference [152] outlines well one of the problems of the WS-Agreement not being based on formal semantics. Example 14 illustrates this problem. Such a construct is valid within WS-Agreement, but requires natural language understanding if automated reasoning is to be used.

```
<QualifyingCondition>
  day of week is a weekday
</QualifyingCondition>
```

Example 14: A valid statement using WS-Agreement

In [152], a technique for binding semantics to a WS-Agreement for use in matching providers and consumers is used. For that work, the ARL rule language[211] and associated agent framework [212] are adopted. However, the approach is only applicable for SLAs defined for web services. Other researchers are working on integrating semantics into the web service framework [144][145], however these projects focus on the service description, not on the agreements for those services.

Flexible communication

QDINE uses an asynchronous, message-based communication mechanism based on the exchange of speech acts. The flexibility attained by such an approach over a simple request/response mechanism as used by the Web Services framework means greater interoperability with other platforms using different communication models and an increase in the potential business models supported. The FIPA standard communication mechanism used by QDINE can support the model adopted by Web Services, in addition to more complex interaction patterns. Such a possibility is made evident by [213] and [214].

The WS-Agreement template

The WS-Agreement template mechanism enables a service provider to publish predefined service agreements to indicate the typical, or allowable agreements that can be created by a consumer. A secondary benefit is the reduction of data transmitted to describe an SLA. To use a predefined template, the SLA template identifier is included in exchanged messages. A WS-Agreement template can describe all or part of a WS-Agreement, such as the service description terms and the guarantee terms. By specifying an SLA template id, only changes from the template are included in messages, relieving the need to describe the entire WS-Agreement in every interaction.

This use of a template introduces two key problems affecting SLAs based on WS-Agreement:

1. The "TemplateID" and "TemplateName" attributes of the WS-Agreement "context" block can take any string value. No formal relationship exists between a template specification and a template id specified on a WS-Agreement. The meaning of the template id can only be assumed when used within the Web Services framework within one particular provider. Consequently, the WS-Agreement is specific to the Web Services framework, thus compromising the re-usability of a WS-Agreement amongst different providers or between the web services framework and any other platforms that may wish to use the WS-Agreement specification.
2. If an SLA is created based on a template, and the template is changed while that SLA is active, then the validity of the SLA using that template is open to dispute. There is no mechanism for a consumer to verify an SLA template associated to an id has not been changed after an agreement is created.

6.4 The Experimental System

An experimental platform was developed using Java based agents to explore the implementation of a system based on the QDINE Negotiation Framework. Java was chosen as there are many open source Java based agent building platforms available, and the use of Java aligns with the author's existing skill.

The goals for developing the experimental system were to:

- Help make assumptions explicit, and discover oversights in functionality provided by the framework.
- Demonstrate the use of OWL-CL within messages sent in a FIPA compliant framework.

6.4.1 The JADE Agent Framework

To avoid building the experimental system from scratch, an existing agent building toolkit was sought. Agentlink[24] maintain a comprehensive list of agent software available, in which a wide range of agent development toolkits are listed. Most agent software is in various research stages with little documentation and a limited user base. There are several large open source and commercial solutions, with Agent Oriented Software's JACK™ product a leader in the commercial space.

A leading community project is JADE[215]; an open source agent platform with a major industry lead from Telecom Italia Lab. JADE is built in line with the FIPA standards, allowing agent interoperability between platforms. It is in active development with a strong user base and a comprehensive set of graphical tools to aid development and debugging. JADE has undergone comprehensive scalability testing [216] and it is shown to scale linearly. Those experiments have shown JADE to support over one thousand agents in a single container.

Using the JADE framework, agent tasks are expressed as "behaviours". Behaviours are logical execution threads that can be composed in various ways to achieve complex execution patterns that can be initialized, suspended and spawned at any given time. An agent core keeps a task list that contains the active behaviours. JADE uses one thread per agent instead of one thread per behaviour to limit the number of threads running in the agent platform. When an agent is born, its `setup()` method is called, following this, a scheduler that is hidden to the developer carries out a round robin policy among all behaviours available in the agent's queue. A behaviour can release its own execution control with the use of blocking mechanisms, or it can permanently remove itself from the queue at run time.

6.4.2 The Jena OWL Application Programming Interface

To manipulate ontological concepts described in OWL, a Java Application Programming Interface(API) was required. The use of a standardised ontology language has enabled the adoption of existing software packages for working with OWL. A number of APIs are available for manipulating OWL files within Java. Five leading ontology APIs were studied for suitability; Jena[217], the Protégé OWL API[218], the WonderWeb OWL API[219], Hawk[220] and Sofa[221]. A summary of the results is presented in Table 6. Jena was chosen because of its support of all OWL features, automated parsing and serialisation of the XML/RDF representation and integrated support for ontology reasoners. Additionally, the Jena API has a smaller memory footprint than its nearest competitor; the Protégé OWL API.

24 http://www.agentlink.org/resources/agent-software.php

RDF/OWL element	Protege OWL API	Jena	Wonderweb API	Hawk	Sofa
(model)	OWLModel	OntModel	OWLOntology	Ontology	OntologyMdl
rdf:Resource	RDFResource	OntResource	OWLEntity	OntObject	Thing
rdf:Property	RDFProperty	OntProperty	OWLProperty (?)	OntProperty	Relation?
-	*OWLProperty*	OntProperty	OWLProperty	OntProperty	Relation
owl:DatatypeProperty	OWLDatatypeProperty	DatatypeProperty	OWLDataProperty	OntProperty	Relation
owl:ObjectProperty	OWLObjectProperty	ObjectProperty	OWLObjectProperty	OntProperty	Relation
-	RDFSClass	OntClass	OWLDescription	OntClass	Concept?
rdfs:Class	RDFSNamedClass	OntClass	?	?	Concept?
owl:Class	OWLNamedClass	OntClass	OWLClass	OntClass	Concept
owl:Class	*OWLAnonymousClass*	(OntClass.isAnon)	?	OntClass	?
owl:Restriction	*OWLRestriction*	Restriction	OWLRestriction	PropertyRestriction	Restriction
owl:allValuesFrom	OWLAllValuesFrom	AllValuesFromRestriction	OWLxAllRestriction	PropertyRestriction	?
owl:someValuesFrom	OWLSomeValuesFrom	SomeValuesFromRestriction	OWLxSomeRestriction	PropertyRestriction	?
owl:hasValue	OWLHasValue	HasValueRestriction	OWLxValueRestriction	PropertyRestriction	Restriction?
owl:cardinality	OWLCardinality	CardinalityRestriction	OWLCardinalityRestriction	PropertyRestriction	Restriction
owl:maxCardinality	OWLMaxCardinality	MaxCardinalityRestriction	OWLCardinalityRestriction	PropertyRestriction	Restriction
owl:minCardinality	OWLMinCardinality	MinCardinalityRestriction	OWLCardinalityRestriction	PropertyRestriction	Restriction
owl:intersectionOf	OWLIntersectionClass	IntersectionClass	OWLAnd	OntClass	?
owl:unionOf	OWLUnionClass	UnionClass	OWLOr	OntClass	?
owl:complementOf	OWLComplementClass	ComplementClass	OWLNot	OntClass	?
owl:oneOf	OWLEnumeratedClass	EnumeratedClass	OWLEnumeration	OntClass	Restriction
(Individual of rdfs:Class)	RDFIndividual	Individual	OWLIndividual	Individual	Thing
(Individual of owl:Class)	OWLIndividual	Individual	OWLIndividual	Individual	Thing
rdf:List	RDFList	RDFList	?		?
owl:Ontology	OWLOntology	Ontology	OWLOntology (?)	Ontology	Ontology
owl:AllDifferent	OWLAllDifferent	AllDifferent	?	Individual	?
owl:DataRange	OWLDataRange	DataRange	?	?	?
rdf:Literal	RDFLiteral	Literal	?	?	Relation
rdf:Resource (untyped)	RDFExternalResource	Resource	?	?	?
owl:AnnotationProperty	isAnnotationProperty()	AnnotationProperty	OWLAnnotationProperty	OntProperty	?
owl:SymmetricProperty	isSymmetric()	SymmetricProperty	isSymmetric	OntProperty	Relation
owl:TransitiveProperty	isTransitive()	TransitiveProperty	isTransitive()	OntProperty	Relation
owl:FunctionalProperty	isFunctional()	FunctionalProperty	isOneToOne()	OntProperty	Restriction??
owl:InverseFunctionalProperty	isInverseFunctional()	InverseFunctionalProperty	isInverseFunctional()	OntProperty	?

Table 6: Comparison of OWL Java APIs[25]

6.4.3 The Agents

Four classes of agent were designed for the experimental system to represent the stakeholders described in section 5.7.2:

1. User Agent (UA) – An end user or organisation is represented by an agent residing at the edge of a managed service zone. The primary goal of the User Agent is to ensure end-user quality of experience. It attempts to achieve this by negotiating with service providers for service level agreements on behalf of the end user or organisation. The designed User Agent performs the QDINE market roles of Subscriber and Consumer.

2. Service Provider Agent (SPA) – The Service Provider Agent represents any provider of a service in the telecommunications value chain. A network operator, network service provider, application service provider, or content provider may

25 Extended and adapted from [222]

employ an SPA to participate in a QDINE market. An SPA performs the QDINE market roles of Subscriber, Consumer and Provider.

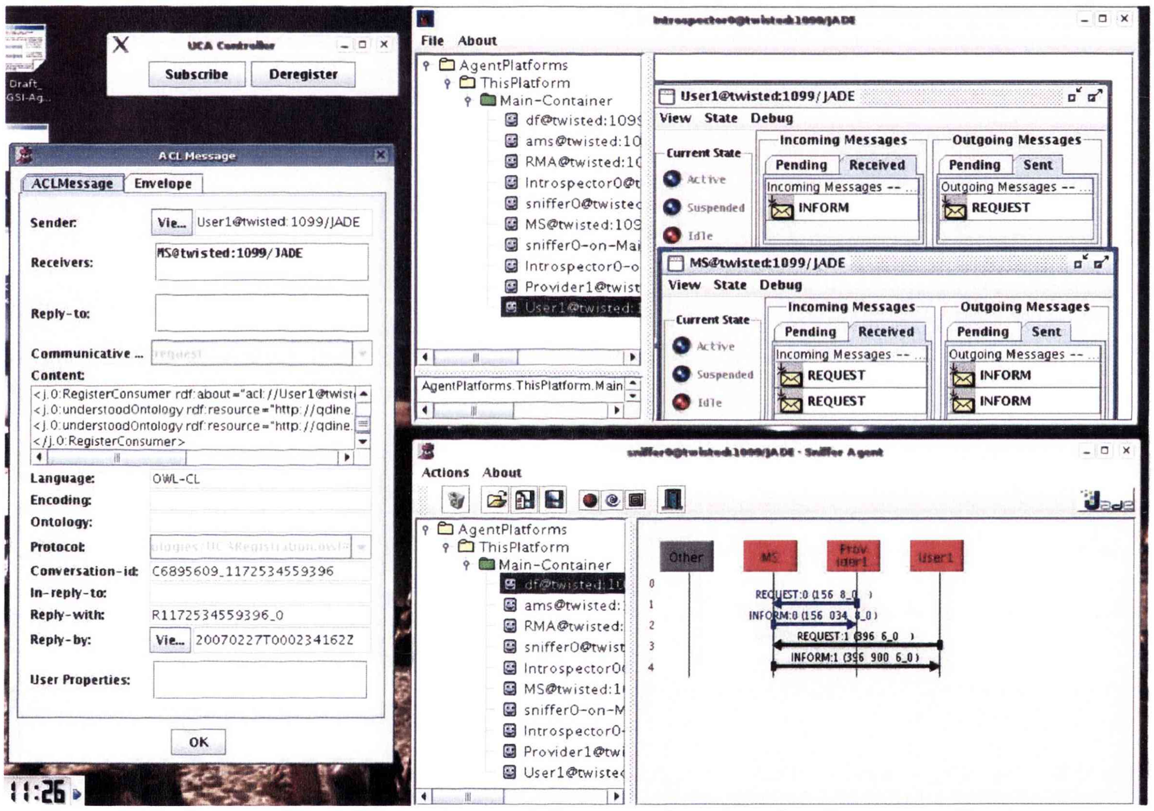
3. Market Agent (MA) – In addition to acting in the “Market Agent” role in a QDINE service market, the Market Agent also performs the subscriber role in order to provide the ubiquitous roaming functionality described in section 5.4.5. Current telecommunication value chains do not typically involve separate entities performing the proposed function of the Market Agent. The Market Agent does not directly represent any particular stakeholder in the current telecommunication value chain. The nearest functionality is performed by operators of bandwidth markets such as Arbinet[91], Band-X[90] and others[89]. As discussed in section 5.7.2.3, a suitable candidate for operating the Market Agent is the Network Operator.
4. Billing Provider Agent (BPA) – The Billing Provider Agent encapsulates the “Billing Provider” role in a QDINE market. The most likely member from the current telecommunications value chain to operate a billing provider is a user's home network service provider. However, a new member of the financial chain may emerge purely as a billing provider. Such an entity may not directly manage service resources, but provide added value to the service delivery chain by providing identification services, and through aggregation and speculation over services from global markets.

6.4.4 Experimental Results

The goals of the experimental framework were to explore the completeness of the QDINE framework functionality, and demonstrate the use of OWL-CL within messages sent in a FIPA compliant framework.

Developing the exploratory system helped to refine the core message exchanges between actors in the framework. The need for the Key Validation and Consumer Authorisation Revocation dialogues (§5.5.2.9 & 5.5.2.6) was discovered through designing and developing the experimental framework.

To explore the use of OWL-CL within FIPA messages, the UCA, SPA and MA were developed and the Consumer and Provider registration dialogues (§5.5.2.1 & 5.5.2.2) were implemented. The initial design and development to enable agents to communicate using OWL was more complex than would be expected if an XML-only approach was taken. However, as anticipated, no system modification was required to parse and validate new message components. The demonstration framework was used to successfully register service providers and consumers with the market agent. Screen captures from the agent registration process are shown in Figure 46. In Figure 46, the upper left "UCA Controller" frame is used to initiate consumer registration and subscription requests from the UCA. The lower left "ACL Message" frame shows a captured registration request message sent by the UCA to the Market Agent. The upper right "Introspector" frame is a component of the JADE toolkit and allows the tracing of behaviours performed by an agent, the messages sent and received and the state of an agent. The lower right "sniffer" frame illustrates the messages sent by agents in the framework as a live sequence diagram.



Sender: User1@twisted:1099/JADE
Receivers: MS@twisted:1099/JADE
Reply-to:
Communicative ...: request
Content:
```
<j.0:RegisterConsumer rdf:about="acl://User1@twist
<j.0:understoodOntology rdf:resource="http://qdine.
<j.0:understoodOntology rdf:resource="http://qdine.
</j.0:RegisterConsumer>
```
Language: OWL-CL
Encoding:
Ontology:
Protocol: ologies/UCARegistration.owl#
Conversation-id: C6895609_1172534559396
In-reply-to:
Reply-with: R1172534559396_0
Reply-by: 20070227T000234162Z
User Properties:

Figure 46: Screenshot of Experimental System, showing the Consumer Registration Message

The preliminary tests performed on the platform indicate that OWL-CL is a suitable content language for use within FIPA messages. This initial work also suggests that the functions provided by the QDINE framework will be sufficient to provide the flexibility in SLA distribution required for delivering new generation services. Extension of the experimental platform to provide full negotiation support is discussed in future work(§7.2).

6.5 The SLA Use Case

An example SLA is now described to explore the application of the SLA ontologies to a specific use. To create the example SLA, the generic SLA ontologies are first extended for a domain specific use.

An SLA obligation includes commitments made by a provider and consumer. Two domain specific ontologies are described to express the commitments made by the provider and consumer on an SLA. The provider commitments are expressed through the service to be provided. In the use case explored, a TEQUILA connectivity SLS is described.

Commitments made by the consumer are expressed using the `hasChargingModel` property defined on a service obligation. For the SLA use case, an event-based monetary charging model is adopted. Finally, to express the outcomes for commitment violations, a violation ontology is described for use within the violation model on a service obligation.

6.5.1 Example Domain-Specific Ontologies

The domain specific ontologies provide an example of how the set of generic ontologies described in chapter 4 can be combined and extended for a domain specific use. Domain specific ontologies should restrict a user of such an ontology to creating instances that "make sense" in its intended context. An example of "not making sense" would be the creation of a `Unit` individual and assigning it to two unrelated classes, such as both a Currency Unit and a Networking Unit unit. This does not "make sense" in the telecommunications context.

The domain specific ontologies are restricted through the use of OWL restrictions on classes and properties, as well as through SWRL rules. Additionally, to ensure logical use of an ontology, some classes receive covering axioms that explicitly define it as a union of one or more subclasses as in Formula 9. Other classes have their allowable individuals restricted to one of an enumerated list. All source files for the domain specific ontology definition files created are listed in Appendix A.3.

$$\textit{Comparator} \equiv \textit{unit:BinaryComparator} \cup \textit{unit:SetComparator} \cup \textit{unit:UnaryComparator}$$

Formula 9: Comparator Type Restriction

6.5.1.1 QDINE SLA Ontology

The QDINE SLA ontology defines a set of domain-specific example subclasses to the classes described in the generic SLA ontology. OWL restrictions are applied to the subclasses to define a set of allowable agreements in a domain specific environment. The QDINE SLA ontology is outline in Figure 47.

For this use case, we a special class of SLA is desired that is non-repudiable, has a validity specified as an `Interval` from the temporal ontology, and places additional restrictions on the obligations used within the agreement. Do achieve this the following was described in OWL:

- A new `QDINE_SLA` class is defined as a subclass of both an `SLA` and `Non-RepudiableThing`.
- A restriction is applied on the `hasValidity` property of `QDINE_SLA`, limiting possible individuals used in the range to the `Interval` class from the Temporal Ontology.
- The `hasObligations` property is restricted to individuals of the `QDINE_-ServiceObligation` class described below.

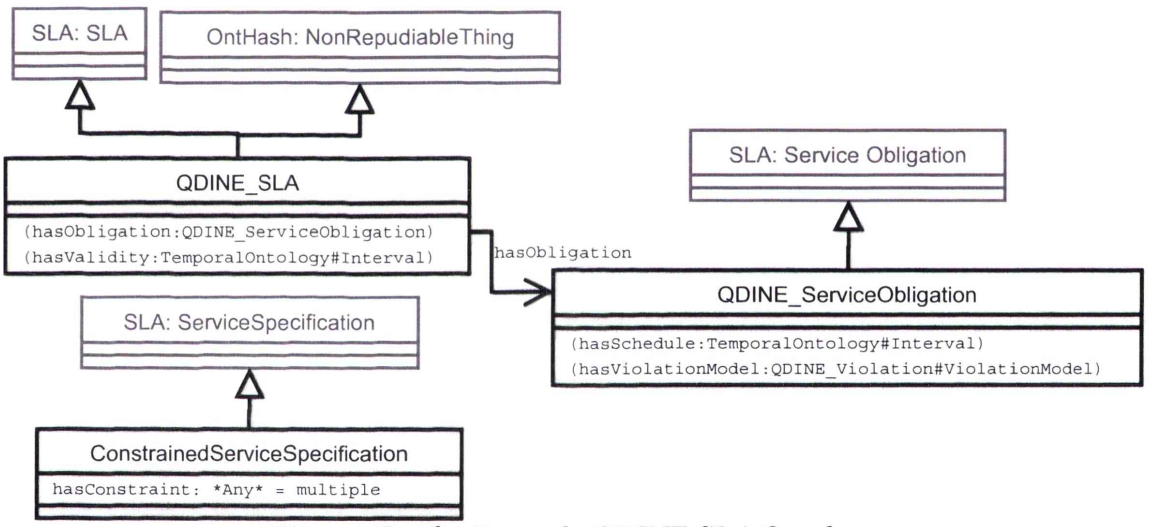

SLA: SLA
OntHash: NonRepudiableThing
QDINE_SLA
(hasObligation:QDINE_ServiceObligation)
(hasValidity:TemporalOntology#Interval)
hasObligation
SLA: Service Obligation
QDINE_ServiceObligation
(hasSchedule:TemporalOntology#Interval)
(hasViolationModel:QDINE_Violation#ViolationModel)
SLA: ServiceSpecification
ConstrainedServiceSpecification
hasConstraint: *Any* = multiple

Figure 47: The Example QDINE SLA Ontology

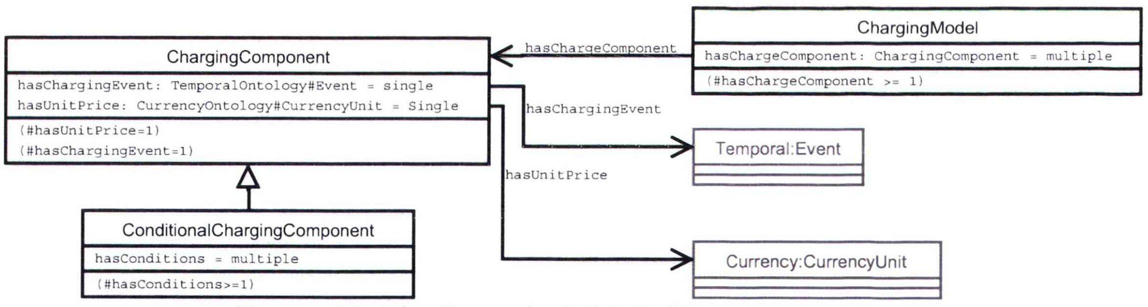
To describe the allowable service obligations included on a QDINE SLA, a new `QDINE_ServiceObligation` class is defined as a subclass of the generic `ServiceObligation` class. For a `QDINE_ServiceObligation`, the `has-ViolationModel` property is restricted to individuals from the `ViolationModel` class from the QDINE Violation Ontology (§6.5.1.4), and the `hasSchedule` property is restricted to individuals from the `Interval` class defined in the `Temporal-Ontology`.

The basic `ServiceSpecification` as defined in the generic SLA ontology is an empty class with no associated properties. The QDINE SLA ontology defines a subclass of the empty `ServiceSpecification` class, named `ConstrainedService-Specification`. A Constrained Service Specification includes a single object property named `hasConstraint`, defining a list of constraints placed on the use of that specification. A constrained service specification can only be used on an SLA if all the constraints are satisfied.

6.5.1.2 QDINE Charging Ontology

No particular charging model is prescribed for the QDINE project, allowing the use of any charging model appropriate to the service context. Additionally, no restriction is made on the `hasChargingModel` property of the `QDINE_ServiceObligation` class to be of a particular type.

An event-based, monetary charging model has been defined and is available for use in SLAs. Event-based charging is used with many current telecommunication services where a charge is applied to the consumer's account each time a specified event occurs. An event may be the beginning of the SLA validity, the passing of a megabyte of data past some measuring interface or a particular time of day. One benefit of using an event-based charging approach is that the QDINE Violation ontology also uses an event-based model. A violation event can therefore be used to trigger a (negative) reimbursement charge. The event-based charging ontology is outlined in Figure 48.


ChargingModel
hasChargeComponent: ChargingComponent = multiple
(#hasChargeComponent >= 1)
hasChargeComponent
ChargingComponent
hasChargingEvent: TemporalOntology#Event = single
hasUnitPrice: CurrencyOntology#CurrencyUnit = Single
(#hasUnitPrice=1)
(#hasChargingEvent=1)
hasChargingEvent
Temporal:Event
hasUnitPrice
ConditionalChargingComponent
hasConditions = multiple
(#hasConditions>=1)
Currency:CurrencyUnit
Figure 48: The Example QDINE Charging Ontology

6.5.1.3 QDINE Entity Ontology

The entity ontology was created to be used with the SLA ontology wherever the details of an entity described on an SLA are required. The ontology is a stub ontology and is a proof of concept more than an ontology that can be used to comprehensively describe entities. The Entity Ontology created is outlined in Figure 49.

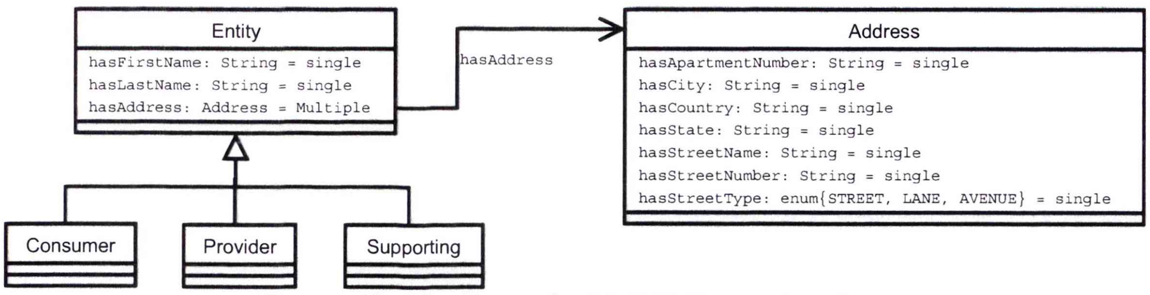

Entity
hasFirstName: String = single
hasLastName: String = single
hasAddress: Address = Multiple
hasAddress
Address
hasApartmentNumber: String = single
hasCity: String = single
hasCountry: String = single
hasState: String = single
hasStreetName: String = single
hasStreetNumber: String = single
hasStreetType: enum{STREET, LANE, AVENUE} = single
Consumer
Provider
Supporting

Figure 49: The Example QDINE Entity Ontology

6.5.1.4 QDINE Violation Ontology

The QDINE Violation Ontology defines a commitment violation model that triggers events in response to the failure of one or more performance levels. Every `ViolationModel` includes one or more `ViolationEvents` that are triggered when the `PerformanceLevel` specified within the violation event evaluates to false.

The violation ontology was created as an example violation model that can be used within the `hasViolationModel` property of a `ServiceObligation`. An advantage of using an event-based model is that the event triggered can be used within other event-based mechanisms such as the QDINE event-based charging mechanism. The violation ontology is outlined in Figure 50.

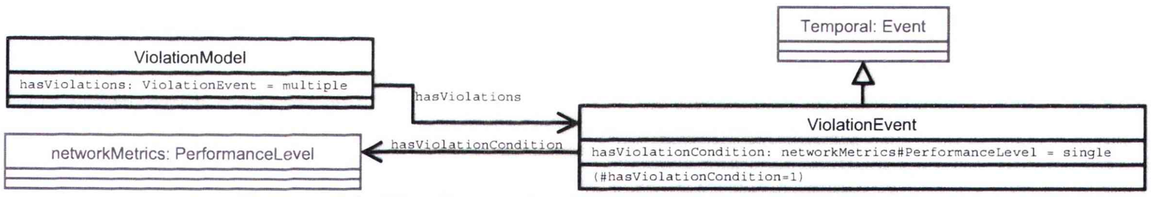

Temporal: Event
ViolationModel
hasViolations: ViolationEvent = multiple
hasViolations
ViolationEvent
networkMetrics: PerformanceLevel
hasViolationCondition
hasViolationCondition: networkMetrics#PerformanceLevel = single
(#hasViolationCondition=1)

Figure 50: The Example QDINE Violation Ontology

6.5.2 Currency Ontology

For completeness, an example currency ontology was created to allow the description of monetary currencies and the relationships between the different units of a currency. Every currency is a subclass of `ContinuousUnit` and SWRL is used in the currency ontology to express the arithmetic relationship between different units of the same currency.

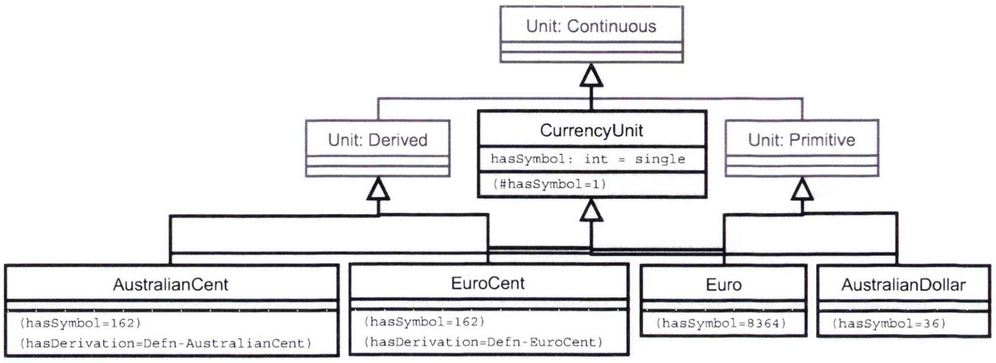
Formula 10 illustrates the relationship between an Australian Dollar and Australian Cent, Where $sameAs(c,d)$ means c and d are the same individual and $multiply(cv,dv,100)$ means $cv = dv * 100$. Similarly, the derivation of `EuroCent` from `Euro` is defined in Formula 11. The currency ontology is outlined in Figure 51.

$$AustralianDollar(d) \wedge AustralianCent(c) \wedge hasFloatValue(d, dv) \wedge hasFloatValue(c, cv) \wedge sameAs(c, d) \Rightarrow multiply(cv, dv, 100)$$

Formula 10: Australian Cent derivation

$$Euro(e) \wedge EuroCent(c) \wedge hasFloatValue(e, ev) \wedge hasFloatValue(c, cv) \wedge sameAs(c, e) \Rightarrow multiply(cv, ev, 100)$$

Formula 11: Euro Cent derivation



SWRL Rules:
Defn-AustralianCent: AustralianDollar(?d) ^ AustralianCent(?c) ^ unit:hasFloatValue(?d, ?dv) ^ unit:hasFloatValue(?c, ?cv) ^ sameAs(?c, ?d) -> swrlb:multiply(?cv, ?dv, 100)
Defn-EuroCent: Euro(?e) ^ EuroCent(?c) ^ unit:hasFloatValue(?e, ?ev) ^ unit:hasFloatValue(?c, ?cv) ^ sameAs(?c, ?e) -> swrlb:multiply(?cv, ?ev, 100)

Figure 51: The Example Currency Ontology

6.5.3 The Use Case – An SLA based on the TEQUILA Service Level Specification (SLS)

6.5.3.1 The TEQUILA SLS

The TEQUILA project [131] created a service specification for DiffServ based connectivity services. This service specification has been well used throughout research [14][60][223] and so has been used here as a comparative example.

To formalise the TEQUILA SLS for use within a QDINE SLA, an OWL ontology has been defined in conformance with the SLS specification presented in [224]. The ontology includes concepts such as `FlowIdentifier`, `Application-Information` and `ExcessTrafficTreatmentAction`, and is outlined in Figure 52 (pg. 177).

6.5.3.2 The Connectivity Service Offered by a Provider

To make services available to consumers within the QDINE framework, a provider publishes service templates to the service registry at the Market Agent. This process is described in sections 5.4.1 and 5.5.2.3. A service template is a `Service-Specification` individual that fully specifies an available service.

For this use case, the service template individual id is `exampleService1` and the service specification class used to create the template is `exampleService1Class`.

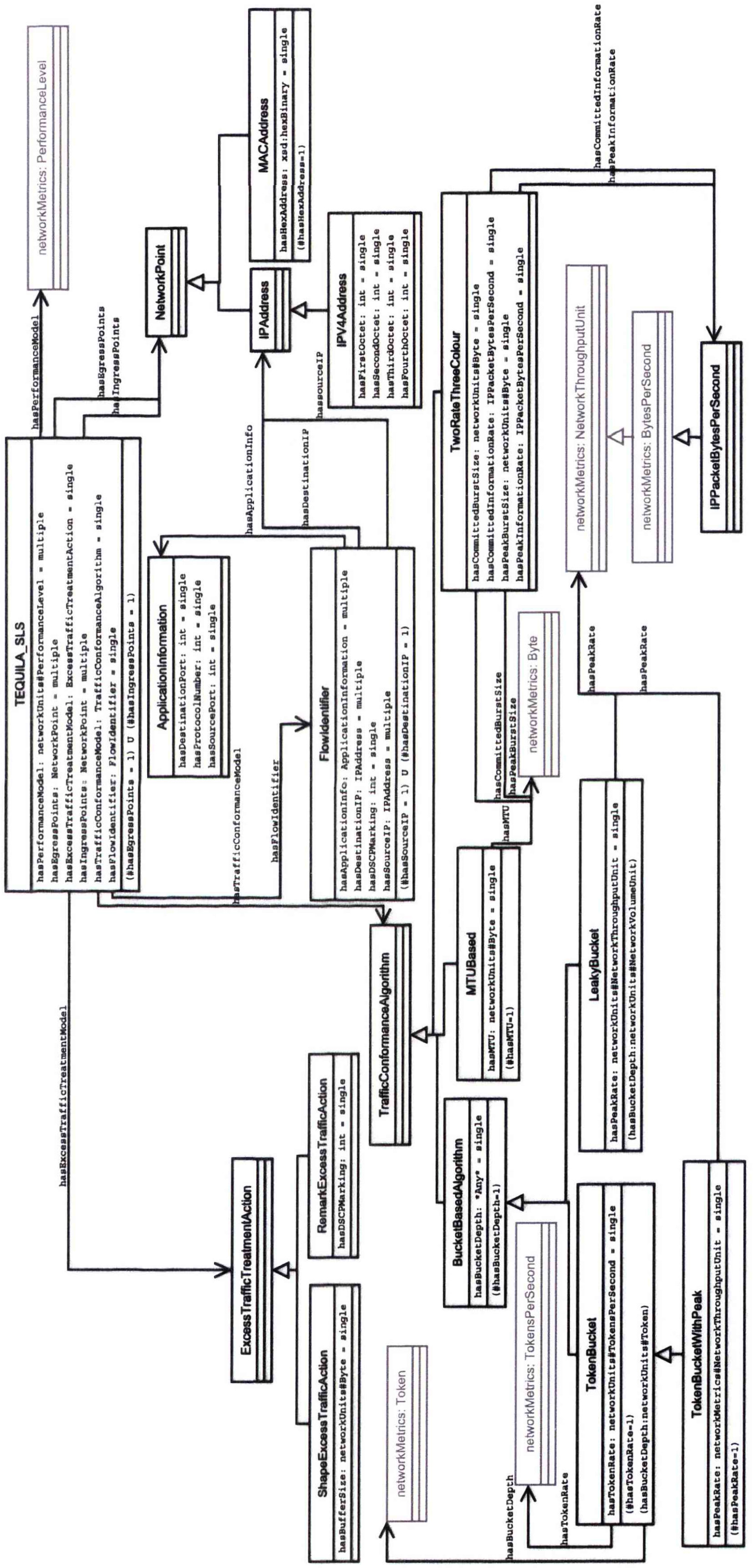

TEQUILA_SLS
hasPerformanceModel: networkUnits#PerformanceLevel = multiple
hasEgressPoints: NetworkPoint = multiple
hasExcessTrafficTreatmentModel: ExcessTrafficTreatmentAction = single
hasIngressPoints: NetworkPoint = multiple
hasTrafficConformanceModel: TrafficConformanceAlgorithm = single
hasFlowIdentifier: FlowIdentifier = single
(#hasEgressPoints = 1) U (#hasIngressPoints = 1)
networkMetrics: PerformanceLevel
NetworkPoint
IPAddress
MACAddress
hasHexAddress: xsd:hexBinary = single
(#hasHexAddress=1)
IPV4Address
hasFirstOctet: int = single
hasSecondOctet: int = single
hasThirdOctet: int = single
hasFourthOctet: int = single
ApplicationInformation
hasDestinationPort: int = single
hasProtocolNumber: int = single
hasSourcePort: int = single
FlowIdentifier
hasApplicationInfo: ApplicationInformation = multiple
hasDestinationIP: IPAddress = multiple
hasDSCPMarking: int = single
hasSourceIP: IPAddress = multiple
(#hasSourceIP = 1) U (#hasDestinationIP = 1)
TwoRateThreeColour
hasCommittedBurstSize: networkUnits#Byte = single
hasCommittedInformationRate: IPPacketBytesPerSecond = single
hasPeakBurstSize: networkUnits#Byte = single
hasPeakInformationRate: IPPacketBytesPerSecond = single
networkMetrics: NetworkThroughputUnit
networkMetrics: BytesPerSecond
IPPacketBytesPerSecond
networkMetrics: Byte
TrafficConformanceAlgorithm
MTUBased
hasMTU: networkUnits#Byte = single
(#hasMTU=1)
LeakyBucket
hasPeakRate: networkUnits#NetworkThroughputUnit = single
(hasBucketDepth:networkUnits#NetworkVolumeUnit)
BucketBasedAlgorithm
hasBucketDepth: "Any" = single
(#hasBucketDepth=1)
TokenBucket
hasTokenRate: networkUnits#TokensPerSecond = single
(#hasTokenRate=1)
(hasBucketDepth:networkUnits#Token)
TokenBucketWithPeak
hasPeakRate: networkMetrics#NetworkThroughputUnit = single
(#hasPeakRate=1)
networkMetrics: TokensPerSecond
networkMetrics: Token
ExcessTrafficTreatmentAction
RemarkExcessTrafficAction
hasDSCPMarking: int = single
ShapeExcessTrafficAction
hasBufferSize: networkUnits#Byte = single
hasPerformanceModel
hasEgressPoints
hasIngressPoints
hasApplicationInfo
hasDestinationIP
hasSourceIP
hasTrafficConformanceModel
hasFlowIdentifier
hasExcessTrafficTreatmentModel
hasCommittedInformationRate
hasPeakInformationRate
hasCommittedBurstSize
hasPeakBurstSize
hasMTU
hasPeakRate
hasBucketDepth
hasTokenRate

Figure 52: The TEQUILA Service Level Specification Ontology

Example 15 outlines the update sent from a provider that is used to publish Example-Service1. The ExampleService1 template individual is created within the provider update, and indicates the service is provided by SPA1. The publication also indicates that ExampleService1 can be provisioned using the bilateral bargaining mechanism defined by example.com: a fictional domain for the purpose of this example.

```
<!-- Defined in the FIPA message ontology: slot
xmlns:qint="QDINE_Interaction.owl#"
xmlns:exserv="exampleService1.owl#"
-->
<qint:ProviderUpdate>
    <qint:hasNewService>
      <exserv:exampleService1Class rdf:ID="exampleService1">
        <qint:provisionedWith>
          <qint:NegotationMechanism
            rdf:resource="http://example.com/kb/al#bilateralBargaining"/>
        </qint:provisionedWith>
      </exserv:exampleService1Class>
    </qint:hasNewService>
    <qint:forProvider>
      <qint:FIPAAgentID rdf:ID="SPA1">
        <qint:hasGUID>provider1@here.org</qint:hasGUID>
      </qint:FIPAAgentID>
    </qint:forProvider>
  </qint:ProviderUpdate>
  <rdf:Description rdf:about="exampleService1.owl#inboundTokenRate">
    <unit:hasFloatValue>1000.0</unit:hasFloatValue>
  </rdf:Description>
```

Example 15: A provider update sent as part of a service publication (§A.4)

Example 16 below provides extracts from ExampleService1Class, including some descriptive annotations between XML comment tags (ie. "<!--...-->"). ExampleService1Class is a subclass of both ConstrainedService-Specification and TequilaSLS, with restrictions placed on the class properties to refine the set of potential class individuals.

```
<owl:Class rdf:ID="exampleService1Class">
  <rdfs:subClassOf
    rdf:resource="QDINE_SLA.owl#ConstrainedServiceSpecification"/>
  <rdfs:subClassOf rdf:resource="TEQUILA_SLS.owl#TEQUILA_SLS"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="TEQUILA_SLS.owl#hasEgressPoints"/>
      <owl:hasValue rdf:resource="ConsumerPoP"/> <!-- An sls:IPv4 address -->
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty
        rdf:resource="TEQUILA_SLS.owl#hasTrafficConfomanceModel"/>
      <owl:hasValue rdf:resource="inboundDataShape"/>
        <!-- An sls:TrafficConformanceAlgorithm -->
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
```

```
      <owl:onProperty
        rdf:resource="TEQUILA_SLS.owl#hasExcessTrafficTreatmentModel"/>
      <owl:hasValue rdf:resource="TEQUILA_SLS.owl#Drop"/>
        <!-- An sls:ExcessTraffictreatmentAction -->
        <!-- Traffic outside the traffic conformance model will be dropped -->
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="TEQUILA_SLS.owl#hasPerformanceModel"/>
      <owl:hasValue rdf:resource="inboundDelayPerformance"/>
        <!-- A metric:PerformanceLevel -->
        <!-- Performance levels are specified for the inbound delay -->
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="TEQUILA_SLS.owl#hasFlowIdentifier"/>
      <owl:hasValue rdf:resource="allInboundTraffic"/>
        <!-- An sls:FlowIdentifier -->
        <!-- This restriction states that the traffic of concern is any -->
        <!-- inbound traffic to the consumer point of presence -->
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="QDINE_SLA.owl#hasConstraint"/>
      <owl:hasValue rdf:ID="volumeChargingConstraint"/> <!-- An swrl:Imp -->
        <!-- This restriction states that individuals of the class can -->
        <!-- only be used within SLAs that use a price per gigabit inbound -->
        >!-- volume charging model -->
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

Example 16: Extract from the exampleService1.owl ontology source file (§A.4)

6.5.3.3 The Created Service Level Agreement

The OWL source for an SLA created from the `exampleService1` template is included below. Some namespaces have been shortened for brevity. The complete source is available in appendix A.4.

```
<qsla:QDINE_SLA rdf:ID="exampleSLA">
  <sla:hasConsumer
    rdf:resource="billingProviderCustomerDatabase.owl#LesGreen001"/>
  <sla:hasValidity>
    <tprl:DiscreteInterval rdf:ID="SLAValidity">
      <tprl:hasStartEvent rdf:resource="exampleService1.owl#startOfContract"/>
      <tprl:hasEndEvent>
        <tprl:DateTimeEvent rdf:ID="SLAFinish">
          <tprl:hasDayOfMonth>1</tmprl:hasDayOfMonth>
          <tprl:hasMonth>1</tmprl:hasMonth>
          <tprl:hasYear>2008</tmprl:hasYear>
        </tprl:DateTimeEvent>
      </tprl:hasEndEvent>
    </tprl:DiscreteInterval>
  </sla:hasValidity>
  <sla:hasObligation>
    <qsla:QDINE_ServiceObligation rdf:ID="IncomingTraffic">
      <sla:hasChargingModel
        rdf:resource="exampleService1.owl#inboundVolumeCharging"/>
      <sla:hasProvider
        rdf:resource="network.provider.com/kb/details.owl#identity"/>
      <sla:hasSchedule rdf:resource="#SLAValidity"/>
      <sla:hasServiceSpecification>
        <exserv:exampleService1Class rdf:ID="lesgreenservice02"/>
      </sla:hasServiceSpecification>
    </qsla:QDINE_ServiceObligation>
  </sla:hasObligation>
  <onthash:hasOntologyHash>
    <onthash:OntologyTreeHash>
      <onthash:forNamespace>exampleservice1.owl</onthash:forNamespace>
      <onthash:hasHashFunction
        rdf:resource="http://www.w3.org/2000/09/xmldsig#sha1"/>
      <onthash:hasHashValue>sdfskjo3524wrlkj</onthash:hasHashValue>
    </onthash:OntologyTreeHash>
  </onthash:hasOntologyHash>
</qsla:QDINE_SLA>

<!-- The following statements further describe components involved with the
     SLA. These statements must be included within the OWL-CL message that
     describes the SLA to ensure they are included in the SLA signature. -->

<rdf:Description rdf:about="exampleService1.owl#pricePerGBinbound">
  <charge:hasUnitPrice>
    <curr:AustralianDollar>
      <unit:hasFloatValue>3.0</unit:hasFloatValue>
    </curr:AustralianDollar>
  </charge:hasUnitPrice>
</rdf:Description>
<rdf:Description rdf:about="exampleService1.owl#ConsumerPoP">
  <sls:hasFirstOctet>192</sls:hasFirstOctet>
  <sls:hasSecondOctet>168</sls:hasSecondOctet>
  <sls:hasThirdOctet>0</sls:hasThirdOctet>
  <sls:hasFourthOctet>1</sls:hasFourthOctet>
</rdf:Description>
<rdf:Description rdf:about="exampleService1.owl#inboundTokenRate">
  <unit:hasFloatValue>1000.0</unit:hasFloatValue>
</rdf:Description>
<rdf:Description rdf:about="exampleService1.owl#startOfContract">
    <tmprl:hasDayOfMonth>1</tmprl:hasDayOfMonth>
    <tmprl:hasMonth>1</tmprl:hasMonth>
    <tmprl:hasYear>2007</tmprl:hasYear>
</rdf:Description>
```

6.6 Summary

Validation of the QDINE work has been performed through a comparison with the closest art, justification of the decisions made while performing the QDINE work, verification that the approach addresses the stated requirements, development of an experimental system as a QDINE proof of concept, and creation of an example SLA to illustrate use of the generic SLA ontologies. The conclusions of this dissertation are now presented in chapter 7.

Chapter 7

Conclusions

The work described in this dissertation is a substantial contribution to the current art. Key components are described below, along with a critical look at the QDINE approach and a set of identified opportunities for future work. Finally, some concluding remarks are made in summary of the dissertation.

7.1 The Key Components

- A new content language to be used within FIPA ACL messages named OWL-CL. OWL-CL is based on the Web Ontology Language (OWL) and is described in section 5.2.2.
- An OWL-based protocol description language that can be used to describe interaction protocols for agent communication. The language is based around a state-transition mechanism for participants, where transitions may include a set of explicitly assigned commitments. The language is described in section 5.2.3.
- A set of ontology definitions that can be used to describe service level agreements for an open range of services in a generalised way. The ontologies include temporal concepts and generic description for units and real-world metrics. These ontologies extend the current art by providing a formal interface to an event calculus and a method to attach non-repudiation information to documents that are based on semantic web technologies. The ontologies are totally independent from any specific management framework, and are described in detail throughout chapter 4.
- A novel negotiation framework to securely manage the service level agreement life-cycle for ubiquitous service delivery. Agent authentication, authorisation, service publication and service discovery are addressed, along with SLA negotiation, binding and cancellation. A significant contribution of the framework is the use of an open market, where context-specific negotiation mechanisms are used to distribute SLAs for available services. The QDINE framework is described in chapter 5.

7.2 Critical Analysis and Future Development of QDINE

A challenge with using formal semantics and a generic framework is the additional effort required to create, and then manipulate the semantic objects. The difference in effort between a generic, semantic approach and domain specific approach where understanding is based on syntax and structure is highlighted in the following two points:

1. The description of domain specific SLA components such as service specifications, charging models and violation models may require a higher initial effort for a generic approach using formal semantics than a domain-specific approach based on syntax and structure. This is because any knowledge required to use each component within a generic approach must be made explicit and semantically associated to the components. Alternatively, a domain-specific approach may use a more simple description of SLA components, as the knowledge required to correctly use the SLA is embedded in the management software and not the SLA description.
   To ease the use of semantic SLA descriptions, there are many tools such as Protégé[167] and Swoop[26] to assist in describing ontologies. Additionally, the containment of knowledge within an SLA specification means less programming changes when a specification is modified.
2. The reasoning required by participants using a generic, semantic approach is typically more computationally intense and complex than a domain-specific approach that processes information using syntax and structure. The complexity problem of using a generic, semantic approach is somewhat compensated by the fact that because the approach is generic, there are existing reasoners available to perform typical operations on knowledge expressed using ontologies. Where a device cannot host a full knowledge reasoner, such capability may be provided by some third party; potentially as a service on a network.

The work described in this dissertation has focussed on two key problems within the larger goal of automated telecommunication-based service management. There is much to be done before such a goal is reached, requiring further work in many research streams. Maintaining a focus on SLA descriptions and SLA management, the following points have either been indicated throughout this dissertation as future work, or are identified as a logical extension to the work already performed:

- To enable automated validation of message conformance to a negotiation protocol, the interface between the QDINE Protocol Description Language (PDL) and the negotiation framework may be extended. The motivation of PDL is to allow automated validation of messages sent as part of a protocol, however this functionality has not been formally verified. The goal of such work would be to

26 http://www.mindswap.org/2004/SWOOP/

ensure all information required to perform such automated validation is included in the protocol description language, and to implement a system to demonstrate the functionality. Work in Electronic Institutions [190] may provide an appropriate background here.

- Within the QDINE framework, the formal specification of negotiation mechanisms used to distribute SLAs has been left for future work. As such, to continue progress towards a functional operating platform, some common negotiation mechanisms such as posted price or a Vickrey auction may be specified.
- The use of grid networks as a platform to deliver high performance computing services is an active and growing research field. Current work in this area has focussed on solving the technological problems for delivering such services, with the business aspect of grid services being largely ignored to date. As grid services mature, adoption by the business community will require support for service contracts and management of business relationships on the grid. The Open Grid Forum (OGF)[225] is primarily concerned with developing a standardised platform for delivering services over a grid network. OGF uses WS-Agreement for describing SLAs, however currently there is no specified solution to managing SLA delivery or lifecycle in accordance with varying business models. A natural extension of the QDINE work is to provide input on WS-Agreement and inclusion of negotiation to the OGF grid architecture.
- To increase interoperability, the ontologies developed in this work may be mapped onto a standard Upper Ontology such as SUMO[200]. As an example of one such mapping, the `Event` class from the QDINE ontologies may be defined as an `owl:equivalentClass` to the `TimePoint` class from the SUMO ontology.
- The termination model for an SLA is introduced in section 4.4.3.4, however because SLA termination may be performed differently for different service contexts, a list of potential termination models is not described. Future work may include the description of example termination models and their interaction with the SLA ontology and negotiation framework. An event-based termination model similar to the QDINE event-based violation model may be appropriate. Termination events may then be used within temporal intervals and the QDINE charging model. Termination events may include supporting dialogues; the outcome of which may be used to trigger the termination event.
- The web services framework is maturing and gaining popularity within the research and business community. Web services can now be used to describe stateful resources, and asynchronous, message oriented communication has recently been proposed. The evolution of web services has involved the adoption of technologies from many fields of research. An example is the use of message based communication, which has been used within multiagent systems for many years.

A natural extension to the QDINE work is the integration of concepts developed into the web services framework. A suitable contribution is the association of context-specific negotiation mechanisms to services. Such functionality is not currently present in Web Services, but will likely be appropriate as web service adoption in the business community increases.

- An increase in billing complexity is expected within a ubiquitous service delivery environment. Seamless, automated billing is therefore a large concern for current and new generation network operators. The way in which a billing path is established need not depend on the service which is being billed, therefore a common technique may be used to establish a billing path for any telecommunication-based service. The QDINE framework includes a very generic process to help "bootstrap" the creation of a billing path between participants, however the range of potential billing mechanisms is not addressed. The interaction of the QDINE framework with some common billing mechanisms may be explored in future work to clarify the SLA/billing interface.
- In section 5.3.3 discussing trust and reputation, the system used to manage agent reputation is left as a concern for future work. Some references to current research are provided, however a detailed discussion is not given. In future work, current research may be analysed for suitability, and either directly adopted or extended for use by agents within QDINE markets.
- In section 5.4.5 discussing service handover, the way in which a consumer in a QDINE market discovers the handover capabilities of peer consumers is left for future work. The use of a distributed peer-to-peer approach may be suitable here, possibly drawing from research in social networks.
- Within this work, an experimental system was developed to test the use of OWL-CL within FIPA-ACL messages, and to help uncover assumptions in the QDINE negotiation framework. A portion of the entire functionality provided by the QDINE framework was developed and tested within the experimental system. Future work within the QDINE project may further develop the experimental system towards a fully functional QDINE framework.

7.3 Concluding Remarks

Telecommunication-based services are an integral component of everyday life. They are used for work, entertainment, personal communication, health and education. The range and scope of services continues to grow, and user access to services is increasing through the increased availability of fixed and mobile networks. Facilitation of future services over ubiquitous new generation networks will require a high level of automation to remain manageable for service users and providers. The proposed QDINE negotiation framework takes an automated, ubiquitous and generalised approach for managing new generation service complexity.

Within telecommunication-based services, the relationship between the provider and consumer of a service is typically a business relationship. QDINE adopts a solution based on well specified service contracts (Service Level Agreements) to handle the dynamic relationships inherent in ubiquitous new generation networks. SLAs specify succinctly a bilateral contract that is specific to a single service instance. To facilitate automated reasoning and interoperability, all components of the solution involved with the communication of information are described using OWL: a standardised Web-based knowledge representation language with a computer-readable syntax and formal semantics.

A wide variety of business models are currently used to provide services, both for telecommunication-based services and services provided through conventional distribution channels. To support a range of business models for services delivered over new generation networks, the QDINE framework is based on an open market design. Any entity can create a market, and within each market, SLAs are formed using market models appropriate to the provided service. The market model to be used for a given service is indicated by the provider when that service is published.

As the closest prior art, the Web Services framework also adopts a somewhat generalised approach to service management. As such, a major strength of Web Services is that any service that is described by the Web Service Description Language can be managed by the framework. Most recently, the benefits of generalised service management have been realised by researchers in grid services. The philosophy of "anything and everything as a service" has emerged within the Open Grid Forum[225]. The delivery of web services or grid services using an open market has not been addressed.

Much work is still required before the entire service delivery life-cycle can be automated, including service agreement, resource configuration and service operation. The QDINE negotiation framework and SLA ontologies described in this work contribute to the service agreement stage in a way that is highly flexible: both in the services that can be managed, and the business models that are used to manage them. This work provides a substantial contribution in its own right, and provides the foundation for a complete solution to this large and complex problem.

INTENTIONALLY BLANK

Appendix A Ontology Source Files

All these ontology source files may be found on the included CD under the /ontologies/ folder.

A.1 The Generic Ontologies

• Ontology Hash Ontology	– File: `/ontologies/OntHash.owl`
• SLA Ontology	– File: `/ontologies/SLAOntology.owl`
• Unit Ontology	– File: `/ontologies/UnitOntology.owl`
• Temporal Ontology	– File: `/ontologies/TemporalOntology.owl`
• Metrics Ontology	– File:`/ontologies/metric.owl`
• Time Unit Ontology	– File: `/ontologies/TimeUnitOntology.owl`
• Network Unit Ontology	– File: `/ontologies/networkUnits.owl`

A.2 Agent Dialogues

• Subscriber Registration	– File: `/ontologies/SubscriberRegistration.owl`
• Consumer Registration	– File: `/ontologies/ConsumerRegistration.owl`
• Provider Registration	– File: `/ontologies/ProviderRegistration.owl`
• Consumer Authorisation	– File: `/ontologies/consumerAuthorisation.owl`
• Billing Authorisation	– File: `/ontologies/BillingAuthorisation.owl`
• Agent Authentication	– File: `/ontologies/AgentAuthentication.owl`

A.3 Example Domain Specific Ontologies

• QDINE SLA	– File: `/ontologies/QDINE_SLA.owl`
• QDINE Charging	– File: `/ontologies/QDINE_Charging.owl`
• QDINE Entity	– File: `/ontologies/QDINE_Entity.owl`
• QDINE Violation	– File: `/ontologies/QDINE_Violation.owl`
• Currency Ontology	– File: `/ontologies/CurrencyOntology.owl`
• TEQUILA SLS	– File: `/ontologies/TEQUILA_SLS.owl`

A.4 OWL Individuals used in Examples

• Example Service 1	– File: `/ontologies/exampleService1.owl`
• Published Template	– File: `/ontologies/publishedTemplate.owl`
• Example SLA	– File: `/ontologies/exampleSLA.owl`
• Example Std. Hash Tree	– File: `/ontologies/hashtrees.owl`

INTENTIONALLY BLANK

Appendix B QDINE Interaction Ontology

The QDINE Interaction ontology defines the vocabulary required to interact within the QDINE framework. It further describes the OWL-CL abstract language concepts such as `Action` and `Proposition` into implementation specific classes such as the *actual* actions performed by agents within QDINE. In addition to extending the abstract components from OWL-CL, the QDINE Interaction ontology also provides an OWL specification for a FIPA-ACL message.

The accompanying diagrams describe the classes in greater detail. Within the diagrams, the first row indicates the class name, the middle rows with multiple columns indicate the properties of the class and the final row indicates any other restrictions or axioms that apply to the class. Within the middle rows, the first column indicates whether the property is a datatype or object property, the second column indicates the name of the property, and the final column is populated with an X if the property is functional (i.i. Can only take one value). The source for the QDINE Interaction ontology is included on the accompanying CD in the location /ontologies/QDINE_Interaction.owl

B.1 QDINE Actions

Each type of standard QDINE action that an agent can perform is described as a separate OWL class, including any parameters appropriate to that action. A fully specified `Action` should include all the information needed for the executor to perform that action.

B.1.1 Action

Within the `QDINE_Interaction` ontology, instances of the `Action` class from OWL-CL are asserted to also be instances of the the `Action` class from the Interaction Protocol ontology. Each action has a status.

OWL-CL#Action
≡ `InteractionProtocolOntology#Action`

B.1.2 Authenticate Agent

AuthenticateAgent		
O	identifiedBy	AuthenticationCredentials

B.1.3 Register Consumer

RegisterConsumer			
O	authenticateUsing	AuthenticationCredentials	X
O	includingAuthorisation	AuthoriseConsumer	X
O	includingserviceSubscription	ProvideServiceUpdates	X
O	supportedNegotiationMechanism	NegotiationMechanism	
O	understoodOntology	owl#Ontology	

B.1.4 Register Provider

RegisterProvider			
O	authenticateUsing	AuthenticationCredentials	X
O	includingConsumerRegistration	RegisterConsumer	X
O	supportedNegotiationMechanism	NegotiationMechanism	
O	understoodOntology	owl#Ontology	

B.1.5 Update Published Services

UpdatePublishedServices			
O	includesUpdate	ProviderUpdate	X

B.1.6 De-register Agent

The de-register agent action is performed by the Market Agent, and can be used to de-register any agent from the market.

DeregisterAgent

B.1.7 Provide Service Updates

ProvideServiceUpdates			
O	AuthenticateUsing	AuthenticationCredentials	X

B.1.8 Authorise Consumer

AuthoriseConsumer			
O	forConsumer	ConsumerDetails	
O	AuthenticateUsing	AuthenticationCredentials	X

B.1.9 Authorise Billing

Before a service provider can bill a billing provider for services used by a customer, the service provider will probably require a billing authorisation from the billing provider.

AuthoriseBilling			
O	forConsumer	ConsumerDetails	
O	revocationAddress	FIPAAgentID	
O	authenticateUsing	AuthenticationCredentials	X

B.1.10 Answer Key Challenge

Used for to verify an agent holds the private key corresponding to some known public key, the process of answering a key challenge involves the performer signing a supplied binary string using its private key and one of the listed signature algorithms. The result of this action is the signed binary string and the signature algorithm used. The signature algorithms available for use are those defined in RFC 4050[226], 4051[227] and those described in the xmldsig specification[188].

AnswerKeyChallenge			
D	challengeString	xsd:base64binary	X
D	signWith	xsd:anyURI	

B.1.11 Provide Service

The `ProvideService` action provides a description of the action of providing some service in accordance with an SLA. The `ProvideService` action includes a property to indicate the SLA that describes the service to be provided.

ProvideService			
O	basedOnSLA		X
#basedOnSLA=1			

B.2 QDINE Propositions

B.2.1 Authentication Revoked

The `AuthenticationRevoked` statement intends to inform the participant that for some reason the authentication provided by the attached certificates is no longer valid.

AuthenticationRevoked		
O	forCertificate	X.509Certificate
#forCertificate>=1		

B.2.2 Consumer Authorisation Revoked

The `ConsumerAuthRevoked` statement is used for an agent to inform another that the consumer authorisation previously granted to the consumers identified has been revoked by the sender of the message.

ConsumerAuthRevoked		
O	forConsumer	ConsumerDetails
#forConsumer>=1		

B.2.3 Billing Authorisation Revoked

The `BillingAuthRevoked` statement allows a billing provider to revoke one or more previously distributed billing authorisations.

BillingAuthRevoked		
O	revokedAuthorisation	BillingAuthorised

B.2.4 Service Changes

The statement of Service Changes describes a set of updates to services available from a set of providers.

IncludesUpdate		
O	includesUpdate	ProviderUpdate
#includesUpdate>=1		

B.2.5 Billing Capability Statement

A `BillingCapabilityStatement` indicates to the receiving party the billing capabilities that the sender supports. These capabilities are defined externally to QDINE, but this statement allows QDINE agents to exchange externally defined billing capabilities.

BillingCapabilityStatement			
O	hasCapability	BillingCapability	

B.2.6 Overridden Knowledge

If an agent has received a piece of knowledge within a message, and that knowledge is in conflict with some existing knowledge, and the receiver has chosen to override the existing knowledge with the new knowledge, the receiver may return an `OverriddenKnowledge` statement to indicate the knowledge that was overridden.

OverriddenKnowledge			
O	overriddenIndividuals		

B.2.7 Action Done

An action done statement is used to inform an agent that some action has been completed successfully. The action done proposition includes the action which has been completed. Within QDINE, there are 7 action-specific "action done" propositions, which also include additional information related to a specific class of completed action. QDINE actions without an associated specific action done proposition can use the generic QDINE action done proposition.

ActionDone			
O	hasAction	owl-cl#Action	
#hasAction>=1			

B.2.7.1 Provider Registered

The `ProviderRegistered` proposition is used to inform the successful completion of a `RegisterProvider` action (§B.1.4). In addition to indicating the completed action, the `ProviderRegistered` proposition enables the market agent to inform a service provider of the market agent's identifying credentials.

ProviderRegistered			
O	AuthenticateUsing	AuthenticationCredentials	X
hasAction(ProviderRegistered,x)⇒ RegisterProvider(x)			

B.2.7.2 Consumer Registered

The `ConsumerRegistered` proposition is used to inform the successful completion of a `RegisterConsumer` action (§B.1.3). In addition to indicating the completed action, the `ConsumerRegistered` proposition enables the market agent to inform a consumer of the market agent's identifying credentials.

ConsumerRegistered			
O	AuthenticateUsing	AuthenticationCredentials	X
hasAction(ConsumerRegistered,x)⇒ RegisterConsumer(x)			

B.2.7.3 Billing Authorised

The `BillingAuthorised` proposition is used to inform the successful completion of an `AuthoriseBilling` action (§B.1.9). It is a non-repudiable legal statement that the sender agrees to accept financial responsibility for all service level agreements signed with the private key corresponding to the public key included on the Authenticated Credentials. In addition to indicating the completed action, the `BillingAuthorised` proposition allows the inclusion of the billing provider's identifying credentials on the returned proposition. Additional constraints on the `BillingAuthorised` statement returned to the provider are expressed below:

$$\begin{gathered}\exists ?bs,?ba : \mathit{BillingAuthorised}(?bs) \wedge \mathit{AuthoriseBilling}(?ba) \wedge \mathit{hasAction}(?bs,?ba) \Rightarrow \\ \exists ?pc,?dc,?cc,?c : \mathit{AuthorisationCredentials}(?pc) \wedge \mathit{DigitalCertificate}(?dc) \wedge \\ \mathit{AuthorisationCredentials}(?cc) \wedge \mathit{ConsumerDetails}(?c) \wedge \mathit{authenticateUsing}(?ba,?pc) \wedge \\ \mathit{forConsumer}(?ba,?c) \wedge \mathit{authenticatedCredentials}(?c,?cc) \wedge (\mathit{X.509Certificate}(?cc) \vee \\ (\mathit{ProxyAuthentication}(?cc) \wedge (\exists ?ct : \mathit{X.509Certificate}(?ct) \wedge \mathit{identifyingCertificate}(?cc,?ct))))\end{gathered}$$

As a non-repudiable legal statement, the `BillingAuthorised` statement is also defined as a `NonRepudiableThing` (§4.4.2.1) and as such must include an ontology hash (§4.4.2) for all the ontologies used within the statement.

BillingAuthorised			
O	AuthenticateUsing	AuthenticationCredentials	X
O	onthash:HasOntologyHash	onthash:OntologyHash	
hasAction(BillingAuthorised,x)⇒ AuthoriseBilling(x)			

B.2.7.4 Agent Authenticated

The `AgentAuthenticated` proposition is used to inform the successful authentication of an agent's proxy credentials, sent within a request to perform an `AuthenticateAgent` action (§B.1.2). By sending this statement, the sender claims that it can positively identify the entity which holds the private key corresponding to the supplied public key. In addition to authenticating the requested agent, the `AgentAuthenticated` statement may include the identification credentials of the billing provider providing the proxy authentication.

AgentAuthenticated			
O	AuthenticateUsing	AuthenticationCredentials	X
hasAction(AgentAuthenticated,x)⇒ AuthenticateAgent(x)			

B.2.7.5 Subscriber Registered

The SubscriberRegistered proposition is used to inform the successful registration of a Subscriber in the service market. In addition to indicating the completed action, the SubscriberRegistered proposition enables the market agent to inform a subscriber of the market agent's identifying credentials.

SubscriberRegistered			
O	AuthenticateUsing	AuthenticationCredentials	X
hasAction(SubscriberRegistered,x)⇒ RegisterSubscriber(x)			

B.2.7.6 Consumer Authorised

This statement is used to inform an agent that an AuthoriseConsumer action has been successfully completed. In the case where a consumer authorisation occurs directly between the consumer and provider, the consumer will likely require the provider to identify itself by returning its identifying credentials.

ConsumerAuthorised			
O	AuthenticateUsing	AuthenticationCredentials	X
hasAction(ConsumerAuthorised,x)⇒ AuthoriseConsumer(x)			

B.2.7.7 Key Challenge Answered

The KeyChallengeAnswered statement indicates that an agent was able to answer the key challenge, and it includes the signed original message string.

KeyChallengeAnswered			
D	signedWith	xsd:anyURI	X
D	responseString	xsd:base64	X
hasAction(KeyChallengeAnswered,x)⇒ AnswerKeyChallenge(x)			

B.2.8 QDINE Reasons

B.2.8.1 Unknown Namespace Prefix

When an agent receives a message of which the content uses a prefix for which the receiver can not determine the associated ontology namespace, the receiver may send the original sender a “not understood” message, including an instance of the following class.

UnknownNamespacePrefix		
D	unknownPrefix	xsd#String

B.2.8.2 Unsupported Namespace

When an agent receives a message on which a namespace is used that is for some reason not supported by the receiver, the receiver may send the message originator a "not understood" message, including an instance of the `UnsupportedNamespace` class, described below.

UnsupportedNamespace		
D	unsupportedNamespace	xsd#anyURI

B.2.8.3 Strict OWL Required

When an agent receives a message in which the `content` slot does not include the opening and closing RDF tags, and the receiver requires strict OWL within a message content, the message originator maybe informed that strict OWL is required within message content. This basic statement has no parameters.

B.2.8.4 Untrusted Signatory

If the receiver of an X.509 Certificate does not trust the signer of a certificate, then it may use this reason to not accept the certificate.

B.2.8.5 Certificate Expired

If an X.509 Certificate has expired, an agent may refuse it, citing this reason.

B.2.8.6 Key Challenge Failed

If an agent challenges another's private key, and the challenge fails, the challenging agent may cite this as a reason for some other related purpose, such as not registering an agent.

B.2.8.7 Authentication Details Required

In any message where some component of the message content contains an unused `authenticateUsing` property, and the message receiver requires authentication details, a reason of `AuthenticationDetailsRequired` may be returned to the message sender.

B.2.8.8 Ontology Hash Mismatch

If a component on a message is instantiated as a `NonRepudiableThing` from the OntHash ontology(§4.4.2.1), and the receiver can not reconcile the hash value against its own generated value for some ontology, the receiver will use the `OntologyHashMismatch` reason including the ontology which cause the mismatch, and the hash values it supports for that ontology URI.

B.2.8.9 Unsupported Role

If an agent ζ receives a message and the participation of ζ in the protocol will require that it sends a message μ, and the act of sending μ is defined as an indication message for some role ϱ, but ζ does not wish to adopt ϱ, then ζ may send a "not understood" message with a reason of `UnsupportedRole`.

B.2.8.10 Message Signature Required

If an unsigned message is sent and the recipient requires that the message is signed, this reason may be returned.

B.2.8.11 Consumer Authorisation Required

If an agent ζ_1 attempts to use some capability from another agent ζ_2 and the use of that capability requires that ζ_1 be authorised from ζ_2 as a consumer, then ζ_2 may send ζ_1 a message including the reason `ConsumerAuthorisationRequired`.

B.2.8.12 Market Authorisation Only

If a service provider only accepts consumer authorisation via the market agent, and an agent other than the market agent attempts to authorise a consumer with the service provider, the service provider may refuse the message citing a reason of `MarketAuthorisationOnly`.

B.2.8.13 Rejected Billing Provider

If an agent attempts to authorise a consumer with a service provider, and the service provider does not accept the billing provider, a `RejectedBillingProvider` reason may be returned, including any extra information in an error message.

B.2.8.14 Unknown RDF Resource

If an agent receives a message that refers to an RDF resource that the recipient is not aware of, this message may be returned, including the list of identifiers that are unknown.

UnknownRDFResource		
D	`unknownResource`	`xsd#anyURI`

B.2.8.15 Consistence Violation

This reason is used whenever information received by an agent would causes the agent's knowledge base to become in an inconsistent state (eg. contradicting knowledge). An agent may with to inform the sender of the violating information, that the information sent has caused the inconsistent state, and what the offending knowledge was.

ConsistenceViolation		
O	`conflictingKnowledge`	

B.2.9 FIPA ACL Message Class

A FIPA ACL message class named `FIPAMessage` is described as part of the `QDINE_Interaction` ontology. The `FIPAMessage` class is a subclass of the OWL-CL `ACLMessage` class(5.2.2.5) and describes the components and restrictions of a FIPA ACL message whilst maintaining strong semantic definition for the slots in the message class. The creation of this class was guided by the FIPA specification for ACL Messages [174].

	FIPAMessage		
O	hasContent	owl-cl#MessageContent	X
O	hasContentEncoding		X
O	hasContentLanguage		X
O	hasContentOntology		
D	hasConversationId		X
O	hasPerformative		X
O	hasProtocol		X
O	hasReceiver		
D	hasReplyBy		X
O	hasReplyTo		
D	hasReplyWith		X
O	hasSender		X
D	inReplyTo		X

```
#hasConversationId = 1
#hasPerformative = 1
#hasProtocol = 1
#hasReceiver >= 1
#hasSender = 1
```

INTENTIONALLY BLANK

Associated Publications

Green, L., 2006, "Service Level Agreements: An Ontological Approach", In Proceedings The Eighth International Conference on Electronic Commerce (ICEC06), August 14-16, Fredericton, Canada, ACM

Green, L., 2006, "Automated SLA Negotiation for Wireless Ubiquity: A Secure Approach", In Proceedings International Conference on Wireless Information Networks and Systems (WINSYS06), August 7-10, Setŭbal, Portugal, INSTICC Press

Green, L., Maknavicius, L., 2006, "Secure Billing for Ubiquitous Service Delivery", In Proceedings Fifth International Workshop on Advanced Internet Charging and QoS Technologies (ICQT06), June 27, St. Malo, France. Also in Lecture Notes in Computer Science (LNCS) Vol 4033, Performability has its Price, pp. 90-101, Springer-Verlag

Green, L., 2004, "Service Level Negotiation in a Heterogeneous Telecommunications Environment", In Proceeding International Conference on Computing, Communications and Control Technologies (CCCT04), August 13-17, Austin, TX, USA

Green, L., 2004, "An Information-Rich, Virtual Trading Environment", In Proceedings Future Business Technology Conference (FUBUTEC04), March 12, INSEAD, Fontainebleau, France, EUROSIS 2004

INTENTIONALLY BLANK

Bibliography

1. Agosto, D., Information literacy: Essential skills for the information age, *Journal of the American Society for Information Science and Technology*, 2005;56:1008-1009.
2. Jafarkhani, H., Space-Time Coding: Theory and Practice, 2005, Cambridge University Press, New York
3. Gabriel, C., A Global View of Pre-WiMAX Deployment, Source: *http://www.wimaxtrends.com/articles/excerpt/e101005a.htm*, Accessed: 2006.03.03
4. The FON project: WiFi Everywhere, Source: *http://en.fon.com/*, Accessed: 2006.03.21
5. T-Mobile first to bridge 3G, EDGE, GPRS, and WiFi, Source: *http://www.cbronline.com/article_news.asp?guid=C0AB6620-5C75-49A9-95B9-0F30F14C3884*, Accessed: 2006.04.05
6. Karp, A. H., Rules of Engagement for Automated Negotiation, *The First IEEE International Workshop on Electronic Contracting (WEC)*, 2004, July, pp. 32-39
7. Bartolini, C., Preist, C., Jennings, N. R., A Software Framework for Automated Negotiation., *SELMAS*, 2004, May, pp. 213-235
8. Giammarino, G., Huard, J., Semret, N., Merkato: a platform for market-based resource allocation, *Intelligent Agents for Telecommunication Environments*, 2000, Hermes Science Publishing Ltd., pp. 77-91
9. Fankhauser, G., Stiller, B., Plattner, B., Arrow: A flexible architecture for an accounting and charging infrastructure in the Next Generation Internet, *Netnomics*, 1999;1:201-223.
10. Yamamoto, L., Leduc, G., Resource Trading Agents for Adaptive Active Network Applications, *Networking and Information Systems Journal*, 2000;3:743-768.
11. Reichl, P., Fankhauser, G., Stiller, B., Dynamic Pricing Schemes for Multi-Class Multi-Provider Internet Connections, CATI, 1999
12. VilÃ, P., Marzo, J. L., Fabregat, R., Harle, D., A Multi Agent Approach to Dynamic Virtual Path Management in ATM Networks, *Agent Technology for Communications Infrastructure*, 2001, John Wiley & Sons Ltd, pp. 167-184
13. Georgoulas, S., Trimintzios, P., Pavlou, G., Joint Measurement- and Traffic Descriptor-based Admission Control at Real-Time Traffic Aggregation Points, *Proceedings IEEE International Conference on Communications (ICC2004), QoS and Performance Symposium*, 2004, June, pp. 816-821
14. Creation and Deployment of End-User Services in Premium IP Networks, Source: *http://wwwcadenus.fokus.fraunhofer.de/*, Accessed: 2004.11.26
15. Kneer, H., Stormer, H., HÃ¤uschen, H., Stiller, B., An Agent-based Framework for Monitoring Service Contracts, *International Conference on Electronic Commerce and Web Technologies (EC-Web 2002)*, 2002, September 2-6, pp. 139-151

16. Molina-Jimenez, C., Shrivastava, S., Crowcroft, J., Gevros, P., On the Monitoring of Contractual Service Level Agreements, *The First IEEE International Workshop on Electronic Contracting (WEC)*, 2004, July, pp. 1-8

17. Cushnie, J., Hutchinson, D., Oliver, H., Evolution of Charging and Billing Models for GSM and Future Mobile Internet Services, *QofIS '00: Proceedings of the First COST 263 International Workshop on Quality of Future Internet Services*, 2000, September, pp. 312-323

18. Electronic Business using eXtensible Markup Language (ebXML), Source: *http:// www.ebxml.org/*, Accessed: 2007.01.16

19. Grigg, I., The Ricardian Contract: A Universal Value Description System, *The First IEEE International Workshop on Electronic Contracting (WEC)*, 2004, July, pp. 25- 31

20. Munro, J, Heroes of the Telegraph, 1997, Project Gutenberg, Gutenberg

21. Pierson O. E., The Western Union Varioplex Telegraph System, *Electrical Communication*, 1944;22:101-109.

22. The Semi-Automatic Ground Environment (SAGE) Project Homepage, Source: *http://www.mitre.org/about/sage.html*, Accessed: 2006.11.01

23. Paul Baran, On Distributed Communications Networks, *First Congress of the Information Systems Sciences*, 1962

24. The RAND Corporation, Source: *http://www.rand.org/*, Accessed: 2006.09.04

25. Rutledge, R. M., Vareha, A. L., Varian, L. C., Weis, A. H., Seroussi, S. F., Mayer J. W., Jaffe, J. F., Angell, M. K., An interactive network of time-sharing computers, *Proceedings of the 1969 24th national conference*, 1969, August, pp. 431-441

26. RFC 36, Protocol Notes, 1970

27. Cerf, V. G., Kahn, R. E., A protocol for packet network interconnection, *IEEE Transactions on Communications*, 1974;COM-22:627-641.

28. RFC675, Specification of Internet Transmission Control Program, 1974

29. Stallings, W., Data and Computer Communications, 2003, Pearson Education, Upper Saddle River, N.J.

30. ITU-T X.25, Interface between Data Terminal Equipment (DTE) and Data Circuit-terminating Equipment (DCE) for terminals operating in the packet mode and connected to public data networks by dedicated circuit, 1996

31. J. W. Burren and Girard and Kirkman, Design for an SRC/NERC Computer network, Rutherford Laboratory, 1977, RL 77-0371A

32. ITU-T I.120, Integrated services digital networks (ISDNs), 1993

33. ITU-T I.233, ISDN frame relaying bearer service, 1991

34. Sigurdson, J., WAP OFF – Origin, Failure and Future, The Stockholm School of Economics, 2001, EIJS WP No.135

35. OECD ICT Key Indicators, Source: *http://www.oecd.org/dataoecd/19/40/34082594.xls*, Accessed: 2006.06.14

36. Bogdanovych, A., Berger, H., Sierra, C., Simoff, S., Narrowing the Gap between Humans and Agents in E-Commerce: 3D Electronic Institutions, *Lecture Notes in Computer Science*, 2005;3590:128-137.

37. Zimmermann, H., OSI Reference Model - The IS0 Model of Architecture for Open Systems Interconnection, *IEEE Transactions on Communications*, 1980;com.28:425-432.

38. ISO/IEC 7498-1, Open Systems Interconnection - Basic Reference Model, 1994

39. Wikipedia OSI Model Table of Examples, Source: *http://en.wikipedia.org/wiki/Osi_model#Table_of_examples*, Accessed: 2006.07.07

40. Oxford English Dictionary, 1989, Oxford University Press

41. Wootton, J, Doctoral Assessment Report - Making Sense of Quality of Experience, University of Technology, Sydney, 2004

42. McKnight, L. W., Bailey, J. P., Internet Economics: When Constituencies Collide in Cyberspace, *IEEE Internet Computing*, 1997;1:30-37.

43. RFC 2474, Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers, 1998

44. RFC 2475, An Architecture for Differentiated Services, 1998

45. RFC 1633, Integrated Services in the Internet Architecture: an Overview, 1994

46. 802.1D-2004, IEEE Standard for Local and Metropolitan Area Networks: Media Access Control (MAC) Bridges, 2004

47. RFC 3031, Multiprotocol Label Switching Architecture, 2001

48. RFC 3036, LDP Specification, 2001

49. RFC 3209, RSVP-TE: Extensions to RSVP for LSP Tunnels, 2001

50. RFC 3630, Traffic Engineering (TE) Extensions to OSPF Version 2, 2003

51. RFC 3784, Intermediate System to Intermediate System (IS-IS) Extensions for Traffic Engineering (TE), 2004

52. RFC 4124, Protocol Extensions for Support of Diffserv-aware MPLS Traffic Engineering, 2005

53. CORBA IIOP Specification, Source: *http://www.omg.org/technology/documents/formal/corba_iiop.htm*, Accessed: 2006.08.04

54. Alcatel-Lucent OmniPCX Enterprise, Source: *http://www1.alcatel-lucent.com/products/productsummary.jsp?productNumber=pcxenterprise_na*, Accessed: 2007.01.18

55. RFC 2096, IP Forwarding Table MIB, 1997

56. RFC 2205, Resource ReSerVation Protocol (RSVP), 1997

57. RFC 1771, A Border Gateway Protocol 4 (BGP-4), 1995

58. Cristallo, G., Jacquenet, C., An Approach to Inter-domain Traffic Engineering, *Proceedings of XVIII World Telecommunications Congress (WTC2002)*, 2002, September, pp.

59. Ho K.H., Wang, N., Trimintzios, P., Pavlou, G., Traffic Engineering for Inter-domain Quality of Service, Centre for Communication Systems Research, University of Surrey, UK, 1998

60. Management of End-to-end Quality of Service Across the Internet at Large, Source: *http://www.mescal.org/*, Accessed: 2004.11.26

61. Nwana, H. S., Software Agents: An Overview, *Knowledge Engineering Review*, 1996;11:205-244.

62. Posegga, J., Karjoth, G., Mobile Agents and Telcos' Nightmares, *Annales des Télécommunications*, 2000;55:29-41.

63. Chess, D. M., Security Issues in Mobile Code Systems, *Lecture Notes In Computer Science*, 1998;1419:1-14.

64. Schoonderwoerd, R., Holland, O.E., Minimal Agents for Communications Network Routing: The Social Insect Paradigm, *Software Agents for Future Communication Systems: Agent Based Digital Communication*, 1999, Springer-Verlag, pp. 305-325

65. Patel, A., Barria, J. A., Pitt, J., IN Load Control Algorithm for Market-Based Multi-Agent Systems, *Agent Technology for Communications Infrastructure*, 2000, John Wiley & Sons Ltd., pp. 249-265

66. Gibney, M.A., Jennings, N.R., Vriend N.J., Griffiths, J.M., Market Based Call Routing in Telecommunications Networks Using Adaptive Pricing and Real Bidding, *Proceedings of the Third International Workshop on Intelligent Agents for Telecommunication (IATA'99)*, 1999, August, pp. 50-65

67. Hansen, M., Jensen, P., Soldatos, J., Vayias, E., Low-Level Control of Network Elements from an Agent Platform, *Agent Technology for Communication Infrastructures*, 2001, John Wiley & Sons Ltd, pp. 156-166

68. Vayias, E., Soldatos, J., Bigham, J., Cuthbert, L., Luo, Z., Intelligent Agents for ATM Network Control and Resource Management: Experiences and Results from an Implementation on a Network Testbed, *Journal of Network and Systems Management*, 2000;8:373-395.

69. Bigham, J., Cuthbert, L. G., Hayzelden, A. L. G., Luo, Z., Flexible Decentralised Control of Connection Admission, *Agent Technology for Communication Infrastructures*, 2001, John Wiley & Sons Ltd, pp. 144-155

70. VilÃ, P., Marzo, J. L., Fabregat R., Harle, D., A Multi Agent Approach to Dynamic Virtual Path Management in ATM Networks, *Agent Technology for*

Communications Infrastructure, 2001, John Wiley & Sons Ltd, pp. 167-184

71. Tennenhouse, D.L., Smith, J.M., Sincoskie, W.D., Wetherall, D.J., Minden, G.J., A survey of active network research, *Communications Magazine, IEEE*, 1997;35:80-86.

72. Tschudin, C., On the Structuring of Computer Communications, 1993, University of Geneva, Switzerland

73. Ascierto, R., Intel: VoIP is beachhead to more collaboration, Source: *http://www.cbronline.com/article_news.asp?guid=C72052C1-9BBB-47B9-8F74-590D96D66F78*, Accessed: 2007.02.27

74. Sevcik, P., The Pitfalls of Scaling VOIP, *Business Communications Review*, 2002;32:8-9.

75. Andersson, A., Ygge, F., Managing Large Scale Computational Markets, *31st Hawaiian International Conference on System Sciences*, 1998, January, pp. 4-13

76. Ygge, F., Akkermans, H., On Resource-Oriented Multi-Commodity Market Computations, *ICMAS 98*, 1998, July, pp. 365-371

77. Ygge, F., Akkermans, H., Andersson, A., Krejic, M., Boertjes, E., The HOMEBOTS System and Field Test: A Multi-Commodity Market for Predictive Power Load Management, *Proceedings of The Fourth International Conference and Exhibition on The Practical Application of Intelligent Agents and Multi-Agents (PAAM 99)*, 1999, April 19-21, pp.

78. Somefun, K., Gerding, E., Bohte, S., La Poutre, H., Automated Negotiation and Bundling of Information Goods, *Lecture Notes in Computer Science*, 2003;3048:1-17.

79. Hoang, D. B., Cohen, H., Cutrell, D., Figueira, S., Lavian, T., Mambretti, J., Monga, I., Naiksatam, S., Travostino, F., DWDM-RAM: An Architecture for Data Intensive Service Enabled by Next Generation Dynamic Optical Networks, *IEEE Globecom 2004, Workshop on High-Performance Global Grid Networks*, 2004, November, pp. 400-409

80. Lavian, T., Hoang, D., Mambretti, J., Figueira, S., Naiksatam, S., Kaushik, N., Inder, M., Durairaj, R., Cutrell, D., Merrill, S., Cohen, H., Daspit, P., Travostino, F., A Platform for Large-Scale Grid Data Service on Dynamic High-Performance Networks, *First International Workshop on Networks for Grid Applications (GridNets)*, 2004, October, pp.

81. Patel A., Prouskas, K., Barria, J., Pitt, J., A Computational Economy for IN Load Control Using a Multi-agent System, *Journal of Network and Systems Management*, 2000;8:397-417.

82. Lazar, A. A., Semret, N., The Progressive Second Price Auction Mechanism for Network Resource Sharing, *8th International Symposium on Dynamic Games and Applications*, 1998, July, pp. 359-365

83. MacKie-Mason, J. K., Varian, H. R., Pricing the Internet, *Public Access to the Internet*, 1995, MIT Press, pp. 269-314

84. Haque, N., Jennings, N. R., Moreau, L., Resource Allocation in Communication Networks Using Market-Based Agents, *Int Journal of Knowledge Based Systems*, 2005;18:163-170.

85. Kneer, H., Häuschen, H., Bauknecht, K., Tradable Service Level Agreements to Manage Network Resources for Streaming Internet Services, *Tenth European Conference on Information Systems (ECIS 2002)*, 2002, June 6-8, pp.

86. Fankhauser, G., Plattner, B., Diffserv Bandwidth Brokers as Mini-Markets, *Workshop on Internet Service Quality Economics*, 1999, Dec. 2-3, pp.

87. Bailey, J., Boroumand, J., The Economics of Internet Differentiated Services: Creating Markets for Leaky Token Buckets, *Fifth INFORMS Telecommunications Conference*, 2000, March 5-8, pp.

88. Dan, A., Ludwig, H., Pacifici, G., Web Services Differentiation with Service Level Agreements, IBM Corp, 2003

89. Bandwidth Market, Source: *http://www.bandwidthmarket.com/*, Accessed: 2005.11.04

90. Band-X, Source: *http://www.band-x.com/en/*, Accessed: 2005.11.04

91. Arbinet, Source: *http://arbinet.com/*, Accessed: 2005.11.07

92. Boutilier, C., A POMDP Formulation of Preference Elicitation Problems, *AAAI/IAAI*, 2002, July, pp. 239-246

93. Dasilva, L. A., Pricing for QoS-Enabled Networks: A Survey, *IEEE Communications Surveys*, 2000;2:1-8.

94. Crawford D. W., Internet services: a market for bandwidth or communication?, *Internet economics*, 1997, MIT Press, pp. 379-400

95. Clark, D. D., Internet cost allocation and pricing, *Internet economics*, 1997, MIT Press, pp. 215-252

96. Key, P., Service differentiation: Congestion pricing, brokers and bandwidth futures, *International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, 1999, June, pp.

97. MacKie-Mason, K., Varian, H. R., Pricing congestible network resources, *IEEE Journal of Selected Areas in Communications*, 1995;13:1141-1149.

98. AC036: DOLMEN - Service Machine Development for an Open Long-term Mobile and Fixed Network Environment, Source: *http://www.cs.helsinki.fi/u/kraatika/DOLMEN/summary.html*, Accessed: 2005.06.21

99. Abarca, C., Farley, P., Forslow, J., Garcia, J. C., Hamada, T., Hansen, P. F., Hogg, S., Kamata, H., Kristiansen, L., Licciardi, C., A., Mulder, H., Utsunomiya, e., Yates, M., Service Architecture, TINA-C, 1997

100. Fankhauser, G., Schweikert, D., Plattner, B., Service Level Agreement Trading for the Differentiated Services Architecture, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology Zürich (ETH), 2000

101. Creation and Deployment of End-User Services in Premium IP Networks, Source: *http://www.cadenus.fokus.fraunhofer.de/*, Accessed: 2004.11.26

102. Lee, J. J., Ben-Natan, R., Integrating Service Level Agreements, 2002, Wiley Publishing, Inc., pp. 44-55, 193-207

103. Decker, S., Melnik, S., Van Harmelen, F., Fensel, D., Klein, M. C. A., Broekstra, J., Erdmann, M., Horrocks, I., The Semantic Web: The Roles of XML and RDF, *IEEE Internet Computing*, 2000;4:63-74.

104. Noy, N. F., McGuinness, d. L., Ontology Development 101: A Guide to Creating Your First Ontology, Stanford Knowledge Systems Laboratory, 2001, KSL-01

105. Extensible Markup Language (XML), Source: *http://www.w3c.org/XML/*, Accessed: 2006.12.08

106. Resource Description Framework (RDF), Source: *http://www.w3c.org/RDF/*, Accessed: 2006.12.08

107. RDF Vocabulary Description Language 1.0: RDF Schema, Source: *http://www.w3.org/TR/rdf-schema/*, Accessed: 2006.12.08

108. OWL Web Ontology Language, Source: *http://www.w3.org/TR/2004/REC-owl-guide-20040210/*, Accessed: 2004.11.26

109. OWL Web Ontology Language Overview, Source: *http://www.w3.org/TR/owl-features/*, Accessed: 2006.12.08

110. The Semantic Web Rule Language (SWRL), Source: *http://www.w3.org/Submission/SWRL/*, Accessed: 2006.12.08

111. Rule Interchange Format, Source: *http://www.w3.org/2005/rules/*, Accessed: 2006.09.06

112. DAML+OIL (March 2001) Reference Description, Source: *http://www.w3.org/TR/daml+oil-reference*, Accessed: 2006.12.08

113. Motta, E., Reusable Components for Knowledge Modelling, 1999, IOS Press

114. Genesereth, M. R., Fikes, R. E., Knowledge Interchange Format, Version 3.0 Reference Manual, Computer Science Department, Stanford University, 1992, Logic-92-1

115. Reed, S. L., Lenat, D. L., Mapping Ontologies into Cyc, *AAAI 2002 Conference Workshop on Ontologies For The Semantic Web*, 2002, July

116. Loom Project Home Page, Source: *http://www.isi.edu/isd/LOOM/*, Accessed: 2006.10.12

117. The Object Management Group (OMG), Source: *http://www.omg.org/*, Accessed: 2006.12.08

118. ISO/IEC 14977:1996, Information technology -- Syntactic metalanguage -- Extended BNF, 1996

119. RDF/XML Syntax Specification (Revised), Source: *http://www.w3.org/TR/rdf-syntax-grammar/*, Accessed: 2006.12.08

120. OWL Web Ontology Language Reference, Source: *http://www.w3.org/TR/owl-ref/*, Accessed: 2006.12.08

121. Bossam Rule/OWL Reasoner, Source: *http://mknows.etri.re.kr/bossam/*, Accessed: 2006.10.13

122. Hoolet, Source: *http://owl.man.ac.uk/hoolet/*, Accessed: 2006.12.08

123. The Rule Markup Language, Source: *http://www.ruleml.org/*, Accessed: 2006.01.17

124. Horrocks, I., Patel-Schneider, P. F., Bechhofer, S., Tsarkov, D., OWL Rules: A Proposal and Prototype Implementation, *Journal of Web Semantics*, 2005;3:23-40.

125. Motik, B., Sattler, U., Studer, R., Query Answering for OWL-DL with Rules, *Proceedings of the 3rd International Semantic Web Conference (ISWC2004)*, 2004, 7-11 Nov

126. Stoutenburg, S., Obrst, L., Orchestration of Ontologies and Rules for Integration of the DoD Enterprise, *8th International Protégé Conference: Protégé with Rules Workshop*, 2005, July 18-21

127. McKenzie, C., Gray, P. M. D., Preece, A. D., Extending SWRL to Express Fully-Quantified Constraints, *Lecture Notes in Computer Science - Rules and Rule Markup Languages for the Semantic Web: Third International Workshop, RuleML 2004, Hiroshima, Japan, November 8, 2004. Proceedings*, 2004;3323:139-154.

128. Preece, A. D., Hui, K., Gray, P., An FDM-based constraint language for semantic web applications, *Lecture Notes in Artificial Intelligence - Agent-Mediated Knowledge Management*, 2004;2926:417-434.

129. FOL RuleML: The First-Order Logic Web Language, Source: *http://www.w3.org/Submission/FOL-RuleML/*, Accessed: 2006.07.14

130. A Proposal for a SWRL Extension towards First-Order Logic, Source: *http://www.w3.org/Submission/SWRL-FOL/*, Accessed: 2006.07.14

131. Traffic Engineering for Quality of Service in the Internet, at Large Scale, Source: *http://www.ist-tequila.org/*, Accessed: 2004.11.26

132. MESCAL Deliverable 1.2 'Initial specification of protocols and algorithms for inter-domain SLS management and traffic engineering for QoS-based IP service delivery and their test requirements', MESCAL, 2004

133. Betge-Brezetz, S., Martinot, O., Marilly, E., Delegue, G., Pro-Active SLA Assurance for Next Generation Network, *World Telecommunication Congress (WTC2002)*, 2002, September

134. Marilly, E., Martinot, O.,Betge-Brezetz, S., Delague, G., Requirements For Service Level Agreement Management, *Proceedings of the IEEE Workshop on IP Operations and Management (IPOM 2002)*, 2002, October

135. Sumesgutner, A., Schmid, P., Hatch, C., Gharib, H., Milham, D., Borg, N., Cselenyi, I., Escudero, S., Cruz, V., Haraszti, P., Kelleher, A., McCarthy, D.,

Specification of Inter-domain Quality of Service Management Interfaces, Euroscom, 2001

136. Farrell, A. D. H., Trastour, D., Christodoulou, A, Performance Monitoring of Service Level Agreements for Utility Computing using the Event Calculus, *The First IEEE International Workshop on Electronic Contracting (WEC)*, 2004, Feb

137. Kowalski, R., Sergot, M, A logic-based calculus of events, *New Generation Computing*, 1986;4:67-95.

138. Skene, J., Lamanna, D. D., Emmerich, W., Precise Service Level Agreements, *Proceedings of the 26th International Conference on Software Engineering*, 2004, May, pp. 179-188

139. UML 2.0 OCL Specification, ptc/03-10-14, 2003 Source: *http://www.omg.org/docs/ptc/03-10-14.pdf*

140. Paschke, A., RBSLA A declarative Rule-based Service Level Agreement Language based on RuleML, *CIMCA '05: Proceedings of the International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce Vol-2 (CIMCA-IAWTIC'06)*, 2005, November, pp. 308-314

141. Web Service Level Agreements (WSLA), Source: *http://www.research.ibm.com/ wsla/*, Accessed: 2004.11.26

142. Sahai, A., Durante, A., Machiraju, V., Towards Automated SLA Management for Web Services, HP Laboratories Palo Alto, 2002

143. Andrieux, A., Czajkowski, K., Dan, A., Keahey, K., Ludwig, H., Nakata, T., Pruyne, J., Rofrano, J., Tuecke, S., Xu, M., Web Services Agreement Specification (WS-Agreement), Global Grid Forum, 2005

144. Top level of OWL ontology for services. Part of the OWL-S ontology, Source: *http://www.daml.org/services/owl-s/*, Accessed: 2006.12.08

145. The Semantic Web Services Framework, Source: *http://www.w3.org/Submission/ SWSF/*, Accessed: 2006.09.10

146. Roman, D., Keller, U., Lausen, H., De Bruijn, J., Lara, R., Stollberg, M., Polleres, A., Feier, C., Bussler, C., Fensel, D., Web Service Modeling Ontology, *Applied Ontology*, 2005;1:77-106.

147. METEOR-S: Semantic Web Services and Processes, Source: *http://lsdis.cs.uga.edu/projects/meteor-s/*, Accessed: 2006.09.10

148. Fensel, D., Bussler, C, The Web Service Modeling Framework WSMF, *Electronic Commerce Research and Applications*, 2002;1:113-137.

149. Miller, J., Verma, K., Rajasekaran, P., Sheth, A., Aggarwal, R., Sivashanmugam, K., WSDL-S: Adding Semantics to WSDL, LSDIS Lab, University of Georgia, 2005

150. Business Process Execution Language for Web Services, Source:

ftp://www6.software.ibm.com/software/developer/library/ws-bpel.pdf, Accessed: 2006.12.01

151. W3C Workshop on Frameworks for Semantics in Web Services Summary Report, Source: *http://www.w3.org/2005/04/FSWS/workshop-report.html*, Accessed: 2006.09.10

152. Oldham, N., Verma, K., Sheth, A., Hakimpour, F., Semantic WS-agreement partner selection, *Proceedings of the 15th International Conference on World Wide Web*, 2006, May 23-26, pp. 697-706

153. Jin, H., Wu, H, Semantic-enabled Specification for Web Services Agreement, *International Journal of Web Services Practices*, 2005;1:13-20.

154. Chieng, D., Ho, I., Marshall, A., Parr, G., An Architecture for Agent-Enhanced Network Service Provisioning through SLA Negotiation, *1st International Conference on Computing in an Imperfect World (Soft-Ware 2002)/2nd European EUNITE Workshop on Computation Intelligence in Telecommunications and Multimedia*, 2002, 8-10 April, pp. 14-30

155. Internet2 Quality of Service, Source: *http://qbone.internet2.edu/*, Accessed: 2006.09.11

156. Ouelhadj, D., Garibaldi, J., MacLaren, J., Sakellariou, R., Krishnakumar, K., Advances in Grid Computing - EGC 2005, , 2005, Springer Berlin / Heidelberg, pp. 651-660

157. Market Managed Multi-service Internet (M3I), Source: *http://www.tik.ee.ethz.ch/~m3i/*, Accessed: 2005.06.21

158. SErvice QUality across Independently managed Networks (SEQUIN), Source: *http://archive.dante.net/sequin/*, Accessed: 2006.09.13

159. P1008: Inter-operator Interfaces for ensuring end to end IP QoS, Source: *http://www.eurescom.de/public/projects/p1000-series/P1008/default.asp*, Accessed: 2006.09.12

160. INTERNODE: INTERworking of NOmadic multi-Domain sErvices, Source: *http://www.mobile-ip.de/~crist/Data/*, Accessed: 2005.06.22

161. AC325 MONTAGE - Mobile Intelligent Agents in Accounting, Charging, and Personal Mobility Support, Source: *http://www.cordis.lu/infowin/acts/rus/projects/ac325.htm*, Accessed: 2005.06.21

162. Telecommunications Information Networking Architecture Consortium, Source: *http://www.tinac.com/index.htm*, Accessed: 2005.05.25

163. Paurobally, S., Jennings, N. R., Protocol engineering for web services conversations, *Engineering Applications of Artificial Intelligence*, 2005;237-254.

164. Waclawsky, J., IMS 101: What You Need To Know Now, *Business Communications Review*, 2005

165. Green, L., Service Level Agreements: An Ontological Approach, *The Eighth International Conference on Electronic Commerce*, 2006, August, pp. 185-194

166. Baader, F., The description logic handbook : theory, implementation, and applications, 2003, Cambridge University Press, Cambridge, UK

167. The Protégé Ontology Editor and Knowledge Acquisition System, Source: *http:// protege.stanford.edu/*, Accessed: 2004.11.26

168. Pinto, H. S., Martins, J. P., Ontologies: How can They be Built?, *Knowledge and Information Systems*, 2004;6:441-464.

169. ISO/IEC 10118-3:2004, Information technology -- Security techniques -- Hash-functions -- Part 3: Dedicated hash-functions, 2004

170. Barreto, P. S. L. M., Rijmen, V., The WHIRLPOOL Hashing Function, *First open NESSIE Workshop*, 2000, 13-14 Nov

171. RFC 3548, The Base16, Base32, and Base64 Data Encodings, 2003

172. Kowalski, R., Sergot, M., A logic-based calculus of events, *New Generation Computing*, 1986;4:67-95.

173. Allen, J. F., Towards a general theory of action and time, *Artificial Intelligence*, 1984;23:123-154.

174. FIPA ACL Message Structure Specification, Source: *http://www.fipa.org/specs/fipa00061/*, Accessed: 2006.02.02

175. FIPA Communicative Act Library Specification, Source: *http://www.fipa.org/specs/fipa00037/*, Accessed: 2006.02.20

176. Toivonen, S., Helin, H., Representing Interaction Protocols in DAML, *Lecture Notes in Computer Science*, 2003;2926:310-321.

177. Desai, N., Mallya, A. U., Chopra, A. K. Singh, M. P., Processes = Protocols + Policies: A Methodology for Business Process Development, Department of Computer Science, North Carolina State University, 2004, TR-2004-34

178. Josephson, W. K. Sirer, E. G., Schneider, F. B., Peer-to-Peer Authentication With a Distributed Single Sign-On Service, *Peer-to-Peer Systems III, Third International Workshop IPTPS 2204*, 2004, February

179. The Java Open Single Sign-On Project, Source: *http://www.josso.org/*, Accessed: 2005.05.25

180. Hellman, M. E., An overview of public key cryptography, *IEEE Communications Magazine*, 2002;40:42-49.

181. Kaliski, B, A Survey of Encryption Standards, *IEEE Micro*, 1993;13:74-81.

182. The Public-Key Infrastructure Charter, Source: *http://www.ietf.org/html.charters/ pkix-charter.html*, Accessed: 2006.04.05

183. X.509, Public-key and attribute certificate frameworks, 2005

184. RFC 1305, Network Time Protocol (Version 3), 1992

185. Communication Security in Multi-Agent Systems, Source: *http://agents.felk.cvut.cz/security*, Accessed: 2006.03.02

186. XML Encryption, Source: *http://www.w3.org/TR/xmlenc-core/*, Accessed: 2006.03.02

187. C S R C - Cryptographic Toolkit: Secure Hashing, Source: *http://csrc.nist.gov/CryptoToolkit/tkhash.html*, Accessed: 2006.04.05

188. XML Digital Signatures, Source: *http://www.w3.org/TR/xmldsig-core/*, Accessed: 2006.03.02

189. Dastani, M., Negotiation protocols and dialogue games, *Proceedings of the Fifth International Conference on Autonomous Agents*, 2001, May, pp. 180-181

190. Esteva, M., Electronic Institutions: from specification to development, 2003, Technical University of Catalonia

191. Pacheco, O., Carmo, J., A Role Based Model for the Normative Specification of Organized Collective Agency and Agents Interaction, *Autonomous Agents and Multi-Agent Systems*, 2003;6:145-184.

192. Sabater, J., Sierra, C., REGRET : a reputation model for gregarious societies, *Fourth Workshop on Deception, Fraud and Trust in Agent Societies*, 2001, May, pp. 61-69

193. Kinateder, M., Rothermel, K., Architecture and Algorithms for a Distributed Reputation System, *LNCS: Trust Management*, 2003;2692:1-16.

194. Buchegger, S., Le Boudec, J., Self-policing mobile ad hoc networks by reputation systems, *IEEE Communications*, 2005;43:101-107.

195. Fähnrich, S., Obreiter, P., Koenig-Ries, B., The Buddy System - A Distributed Reputation System Based On Social Structure, *First Intl. Workshop on Data Management in Mobile Environments*, 2004

196. Morand, P., Boucadair, M., Coadic, T., Levis, P., Egan, R., Asgari, H., Griffin, D., Griem, J., Spencer, J., Trimintzios, P., Howarth, M., Wang, N., Flegkas, P., Ho, K.-H., Georgoulas, S., Pavlou, G., Georgatsos, P., Damilatis, T., Mykoniati, E., Liabotis, I., D1.3: Final specification of protocols and algorithms for inter-domain SLS management and traffic engineering for QoS-based IP service delivery, MESCAL, 2005

197. Nielsen, J., Designing Web Usability: The Practice of Simplicity, 2000, New Riders Publishing, Indianapolis

198. Ontosem, Source: *http://morpheus.cs.umbc.edu/aks1/ontosem.owl*, Accessed: 2007.01.10

199. The AKT Reference Ontology, Source: *http://www.aktors.org/ontology/support*, Accessed: 2007.01.10

200. The Suggested Upper Merged Ontology, Source: *http://reliant.teknowledge.com/ DAML/SUMO.owl*, Accessed: 2007.01.10

201. Mid-level Ontology, Source: *http://reliant.teknowledge.com/DAML/Mid-level-ontology.owl*, Accessed: 2007.01.10

202. The Foundation for Intelligent Physical Agents, Source: *http://www.fipa.org/*,

Accessed: 2007.02.21

203. XC00011B, FIPA RDF Content Language Specification, 2001

204. FIPA OWL Content Language and ACL representation used for the TAGA competition, Source: *http://taga.sourceforge.net/owl/fipaowl.owl*, Accessed: 2006.01.16

205. Zou, Y., Finin, T., Ding, L., Chen, H., Pan, R., Using semantic web technology in multi-agent systems: a case study in the TAGA trading agent environment, *ICEC '03: Proceedings of the 5th international conference on Electronic commerce*, 2003, pp. 95-101

206. Benyoucef, M., Keller, R. K., An Evaluation of Formalisms for Negotiations in E-commerce, *Workshop on Distributed Communities on the Web*, 2000, June, pp. 45-54

207. Dolog, P., Model-Driven Navigation Design for Semantic Web Applications with the UML-Guide, *Engineering Advanced Web Applications: Proceedings of Workshops in connection with the 4th International Conference on Web Engineering (ICWE 2004)*, 2004, 28-30 July, pp. 75-86

208. The ELENA ontology for finite state machines, Source: *http://www.l3s.de/~dolog/fsm/fsm.owl*, Accessed: 2006.02.16

209. OASIS Web Services Resource Framework (WSRF) TC, Source: *http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrf*, Accessed: 2007.01.04

210. Web Services Description Language (WSDL) 1.1, Source: *http://www.w3.org/TR/ wsdl*, Accessed: 2007.01.04

211. Introduction to Able Rule Language (ARL), Source: *http://www.research.ibm.com/able/doc/reference/com/ibm/able/rules/doc-files/arlIndex.html*, Accessed: 2007.01.04

212. Agent Building and Learning Environment (ABLE), Source: *http://www.research.ibm.com/able/*, Accessed: 2007.01.04

213. Greenwood, D., Calisti, M., Engineering Web Service - Agent Integration, *IEEE International Conference on Systems, Man and Cybernetics*, 2004, October 10-13

214. Nguyen, X., Kowalczyk, R., Chhetri, M., Grant, A., WS2JADE: A Tool for Run-time Deployment and Control of Web Services as JADE Agent Services, *Software Agent-Based Applications, Platforms and Development Kits*, 2005, Birkhäuser Basel, pp. 223-251

215. Java Agent DEvelopment Framework (JADE), Source: *http://jade.tilab.com/*, Accessed: 2007.01.07

216. Cortese, E., Quarta, F., Vitaglione, G., Scalability and Performance of JADE Message Transport System, Telecom Italia Lab, 2002

217. Jena Semantic Web Framework, Source: *http://jena.sourceforge.net/*, Accessed: 2007.01.10

218. protégé-owl api, Source: *http://protege.stanford.edu/plugins/owl/api/*, Accessed: 2007.01.10

219. WonderWeb API, Source: *http://wonderweb.man.ac.uk/owl/*, Accessed: 2007.01.10

220. HAWK: OWL Repository and Toolkit, Source: *http://swat.cse.lehigh.edu/projects/index.html#hawk*, Accessed: 2007.01.10

221. Simple Ontology Framework API (SOFA), Source: *http://sofa.projects.semwebcentral.org/*, Accessed: 2007.01.10

222. protégé-owl api, Source: *http://protege.stanford.edu/plugins/owl/api/*, Accessed:

223. AQUILA: Adaptive Resource Control for QoS Using an IP-based Layered Architecture, Source: *http://www-st.inf.tu-dresden.de/aquila/*, Accessed: 2005.06.21

224. Goderis, D., Van Den Bosch, S., T'joens, Y., Poupel, O., Jacquenet, C., Memenios, G., Pavlou, G., Egan, R., Griffin, D., Georgatsos, P., Georgiadis, L., Van Heuven, P., Service Level Specification Semantics and Parameters, IETF, 2002, draft-tequila-sls-02

225. Open Grid Forum, Source: *http://www.ogf.org/*, Accessed: 2007.01.24

226. RFC 4050, Using the Elliptic Curve Signature Algorithm (ECDSA) for XML Digital Signatures, 2005

227. RFC 4051, Additional XML Security Uniform Resource Identifiers (URIs), 2005