Towards a Precise Understanding of Service Properties

by
Justin O'Sullivan
B. Applied Science(Computing)
M. Applied Science(Computing)

Principal Supervisor: Dr. David Edmond
Associate Supervisor: Associate Professor Arthur H.M. ter Hofstede

A dissertation presented to the
Faculty of Information Technology
Queensland University of Technology
in fulfilment of the requirements for the degree of
Doctor of Philosophy

Keywords

non-functional properties, service description, web services, service taxonomy, service semantics

Abstract

This thesis addresses the question of what would be a domain independent taxonomy that is capable of representing the non-functional properties of conventional, electronic and web services. We cover all forms of services, as we prefer not to make any distinction between the three forms. Conventional service descriptions, such as newspaper advertisements, are rich in detail, and it is this richness that we wish to make available to electronic and web service descriptions. In a conventional service context, when we ask a service provider for details, perhaps by phoning the service provider, we are seeking ways to assist with decision making. It is this same decision making or reasoning that we wish to be available to electronic services. Historically, services have always been distinguished according to some criteria of a service requestor. Examples are price, payment alternatives, availability and security. We are motivated to ensure that the criteria used to evaluate conventional services are also available for electronic and web services. We believe that the ability to richly and accurately describe services has significant applicability in the areas of electronic service discovery, dynamic service composition, service comparison, service optimisation, and service management. In particular, the increased level of descriptive depth will also facilitate more thorough decision-making by a service requestor. Whilst we acknowledge the importance of service functionality, this thesis is primarily concerned with the non-functional properties of services. A service is *not* a function alone. It is a function performed on your behalf at a cost. And the cost is not just some monetary price; it is a whole collection of limitations. This thesis is all about these. We believe that to accurately represent any service, a description requires information relating to both the functionality and the associated constraints. We consider these constraints over the functionality of the service to be non-functional properties. We believe that a service description is only complete once the non-functional aspects are also expressed. We undertook a significant analysis of services from numerous domains. From our analysis we compiled the non-functional properties into a series of 80 conceptual models that we have categorised according to availability (both temporal and locative), payment, price, discounts, obligations, rights, penalties, trust, security, and quality. Our motivation is to provide a theoretical basis for automated service discovery, comparison, selection, and substitution. The need to describe a service is analogous with labelling for goods or products. Product labelling occurs for the safety and benefit of purchasers. Why is the same labelling not afforded for the benefit of service requestors?

For Kath. With all the love my heart has to give.

Contents

List of Figures

List of Tables

Listings

Statement of Original Authorship

The work contained in this thesis has not been previously submitted to meet requirements for an award at this or any other higher education institution. To the best of my knowledge and belief, the thesis contains no material previously published or written by another person except where due reference is made.

Signature

Justin O'Sullivan

Date

Acknowledgements

Whilst a PhD is intended to be a contribution by one person, it is really the combined efforts of a number of people. In my case, this PhD journey has been guided and significantly contributed to by two quintessential gentleman - David Edmond and Arthur ter Hofstede.

Gents, this has been an unbelievable six and half year journey. I will forever be indebted to you both for your friendship, support, advice, availability and encouragement. I thank you sincerely for proactively extending support to me during a low point in my research. The simple act of offering to meet after hours, at a time when I was about to wave the *white flag*, enabled me to continue with renewed hope. I am conscious of the subtlety of your efforts, to focus on a specific research topic, yet you were really teaching me to critically think in a more general context. I would not have managed to complete this research without your assistance.

As I move on, I will look back on our regular meetings with both laughter and frustration. David, I thank you for teaching me the value of three simple words - "I don't know". Often I would attend meetings where discussions ended with those three words. Prior to doing this PhD, uttering "I don't know" felt like I was admitting failure of some sort. Now, it makes me think that there is something new to go away to learn or discover. Arthur, I thank you for opening my eyes to the beauty of conceptual modelling. I'm also amazed by your uncanny ability to map parts of my research to hilarious Dutch comedy skits! I think most of all I will look back with appreciation and gratitude for the six and half year commitment that you both gave to help me achieve a very important life goal. Considering this was a team effort, I feel that I personally received more out of this journey.

Looking back, the thought of completing this journey without the support of my lovely wife, Kathy, is near to unimaginable. It's only now though, in contemplation, that I realise the extent of her support during this time. Whilst I tried to complete as much of this work as possible without impacting family life, there have been times when it has had an impact. Kath, I thank you for understanding the importance of this personal journey to me. I appreciate the times when you realised that I needed some space to get on top of it. You are an amazingly giving person and I love you dearly! My PhD journey (our PhD journey!) has undoubtedly intermingled with our own life journey. The birth of our three beautiful children during this time has been a wonderful extension of the love that we share. To Nathan, Chloe and Hayden, thank you for reminding me daily about what is truly important in life. I hope that you will know the fulfilment of dreams in your own lives.

To my Mum and Dad, Vivienne and Peter, I want to say thank you for giving

me the opportunity to study. It has fostered what I believe will be a lifelong love of learning. I also appreciate tremendously the support that you have provided in many ways during my research.

To my father-in-law Robert, I want to say thank you for taking an interest in my research. It was always great to talk to someone who understands the journey through having completed a PhD yourself.

Part of this research was conducted in the context of the REDCONE project, an Australian Research Council SPIRT Grant entitled "Self-describing transactions operating in a large, heterogeneous, distributed environment". This project was a collaboration between GBST Holdings Pty Ltd, the Queensland University of Technology, and the University of New South Wales. I am grateful for the support provided to me by the grant and GBST.

Justin O'Sullivan
October 2006
Queensland University of Technology

Chapter 1

Introduction

Services are ubiquitous. We encounter them in our everyday life. Yet, for all their ubiquity, no standard currently exists that is capable of accurately representing them. The need to describe a service is analogous with the requirement for labelling goods or products. Product labels provide a summary description of the goods to which they are attached. Prospective buyers use the information, together with the price, to make a rational decision about purchasing the product. The label on a jar of pasta sauce may contain the product name, the type of sauce, the manufacturer's name, their address, the ingredients, a bar code, an expiry date, a date of manufacture or batch number, directions for use, benefits or drawbacks (i.e. nutritional information), customer feedback details (e.g. a phone number) and possibly even examples for use (e.g. a recipe). Product labelling occurs for the safety and benefit of purchasers. Why is the same labelling not afforded for the benefit of service requestors?

1.1 Problem area

Let us consider some service related questions. When you encounter a service how do you determine how to request it? What is the identity of the service provider? Where and when is the service available? Can you trust the provider of the service? What quality of service can you be guaranteed? What payment options are available? Does the provider offer any type of discounts? What rights as a service requestor do you have over the service? What other services does this service provider use in providing their service? Once requested, what are the models of interaction that might occur during its delivery?

Whilst a service requestor can ask a service provider for these details in a conventional service context (e.g. by phoning the service provider), electronic representations for services currently lack the descriptive depth to answer these types of questions. This limits the ability to perform a level of semi-automated reasoning over the service description. Conventional service descriptions (e.g. advertisements in a newspaper) are rich in detail, and it is this richness that we wish to make available within electronic service descriptions. The lack of descriptive depth in electronic service descriptions is a result of:

- The lack of basic concepts for describing services. For example, the name given to the style of pricing applied to a service by a service provider.

- The heterogeneous nature of services. Although some services are similar (e.g. taxiing people between locations), some service providers will specialise with a certain style of taxi (e.g. a limousine).

- Existing approaches to service description do not give appropriate consideration to the context that a requestor brings to the service discovery and invocation. For example, if we query the Yellow Pages [121] for a theme park entertainment service we cannot determine whether the service offers a discount for students or pensioners.

- The inherent complexity of services. For example, the difficulty of describing a complex service may hinder a provider from describing a service in satisfactory detail. Arguably, this difficulty with description inhibits the commoditisation of services as requestors are unable to compare services and match their functional and non-functional properties. Additionally, some service providers may want to limit a requestor's ability to compare services based on lower-order properties (e.g. price), as well the eagerness of vendors to provide solutions that meet requestors' needs (i.e. they wish to appear flexible to the service requestor).

We believe that to accurately represent any service, a description requires information relating to both the functionality and the associated constraints. We also believe that the ability to richly and accurately describe services has significant applicability in the areas of electronic service discovery, dynamic service composition, service comparison, service optimisation, service evolution and service management. In particular, the increased level of descriptive depth will also facilitate more thorough decision-making by a service requestor. The current inability to accurately represent services hinders the dynamic opportunities available to organisations via electronic service orchestration.

This research has been motivated by the everyday services that surround us, and the ways in which we engage with them. We believe that the historical interactions, both social and economic, of conventional services offer strategic insight for the success of electronic service initiatives.

Through media such as newspapers, letterbox flyers, corporate brochures and television we are regularly confronted with descriptions for conventional services. These representations vary in the terminology utilised, the depth of the description, the aspects of the service that are characterised and their applicability to candidate service requestors. Existing service catalogues (such as the Yellow Pages) provide little relief for service requestors from the burden of discovering, comparing and substituting services. Add to this environment the rapidly evolving area of web services with its associated surfeit of standards, and the result is a considerably fragmented approach to the description of services.

Before discussing our concerns with existing approaches to description, let us first consider what is involved. The act of describing is performed prior to advertising. This simple fact provides an interesting paradox as services cannot be described exactly before advertisement. Therefore services can't be advertised exactly. This doesn't mean they can't be described comprehensively. By "exactly", we are referring to the fact that context provided by a service requestor (and their service needs) will alter the description of the service that is presented during discovery. For example, suppose a cinema operator wants to describe the price of its service. Let's say the advertised price is $15. They also want to state that a pensioner discount and a student discount is available which provides a 50% discount. Customers (i.e. a service requestor) uses the cinema web site to purchase tickets online. They find the movie of their choice at a time that suits. However, its not until some context is provided by the requestor that the exact price is determined. The requestor might state that they are a pensioner. The same is applicable for a service requestor who purchases multiple tickets perhaps on behalf of other people. The disconnect between when the service is described and when a requestor provides context introduces challenges to the description process. A service provider would be ill-advised to offer independent descriptions that represent all the permutations possible for a single service. The descriptive effort would be prohibitive.

Other than proprietary service catalogues (e.g. Yellow Pages [121]), web service standards such as [23, 13, 47] are the basis for the majority of existing description efforts. We propose some sample questions that we put to existing web service description standards as a means of highlighting their limitations:

1. What is the length of period allowed for deferred payment, and what is the minimum payment required to use a deferred payment option?

2. What percentage deduction does the service provider offer when price matching?

3. What trademark or patent intellectual property rights exist with respect to a service?

4. How many reward scheme points are required for redemption with a particular service?

5. What rights with respect to suspension/resumption, warranty, extension or privacy does a service offer for its requestors?

Our concerns with the existing approaches to web services, and in particular, web service description are two-fold. Firstly, we believe that conventional services are being ignored for a purely web services view of service description. We refer to this as "web service tunnel vision" [72]. Are web services so different from conventional services? Amongst other usages, its our belief that web services will act as the electronic request mechanism for conventional services. Take, for example, a web service that allows flight/travel bookings. Under this scenario, the exposed web service becomes an enabler of automation. We believe that there are numerous

benefits for removing the "tunnel vision" of service descriptions to include both web and conventional services. These benefits include:

1. The ability to perform discovery that returns candidate services from both types of services. This implies that service catalogues would be capable of storing both types of service descriptions.

2. Consistency of terminology between conventional and web services provides protection to service requestors.

To achieve these benefits, a comprehensive service description technique is required. That technique must be capable of expressing the functional and non-functional aspects of both conventional and electronic services. We subscribe to the notion that non-functional properties are constraints over the functionality [24]. This type of technique brings with it a certain series of challenges. Foremost is the need to support service providers in the description of their services in ways similar to their current approach. For example, a service provider can describe the temporal availability of its service using a recurring time interval (e.g. every Monday to Friday from 9am - 5:30pm). We consider such a technique to enable "rich" descriptions. Some implications are:

1. Richly described services increase the complexity of user interfaces for discovering services. It introduces the opportunity to include additional filter criteria into the user interface to assist the discovery process.

2. Richly described services require more effort on behalf of the service provider to describe its service (although varying depths of description would be available). There would be a need to entice service providers to describe their services as richly as possible. In conjunction with a single standardised technique, there is the possibility for reuse of service descriptions between service catalogues.

3. A language that is flexible may make the discovery process more difficult. For example, a temporal interval may be expressed as specific set of dates/times, or it could be expressed as an overall interval with restrictions limited to sub-intervals.

Our second concern with respect to existing service description techniques is that the semantic richness of the non-functional properties of services is not being exploited. We have previous referred to this as "semantic myopia" [72]. Our previous work has highlighted the need to describe the non-functional properties of services, as well as the types of properties that should be described [68].

Our work is an attempt to narrow the void between the functionally focused web service description standards and the non-functional description of services [70]. There we present a discussion of many non-functional properties (mentioned in section 1.4). The intent is to outline a basic set of domain independent non-functional properties that can be used to improve discovery, comparison and service substitution. It should be noted that it is almost impossible to provide a suitable taxonomy

for every domain to which services might apply (e.g. from pathology to plumbing). It might however be possible to provide a taxonomy of the remaining concepts (i.e. the "non-functional properties") and this is what this thesis attempts to establish. The approach we've used has provided the opportunity to augment our work with description techniques such as the Web Services Modelling Ontology (WSMO) [82], or equally it can be considered a separate concrete language. Without rich service descriptions we are incapable of achieving automated service discovery, comparison, negotiation and substitution.

1.2 Succinct research question

The basic question that this thesis attempts to answer is what would be a domain independent taxonomy that is capable of representing the non-functional properties of services for conventional, electronic, and web services.

1.3 Criteria for a solution

It is proposed that the solution will address a series of criteria. Whilst the criteria address specific requirements in service description, they also address more general requirements of information modelling [99] and knowledge representation [30]. These criteria will be used to evaluate the robustness of the solution. The criteria are:

- **Criterion #1: The service description language must support decision-making with respect to services.** To support any level of decision-making we believe that the service descriptions should have a *formal* grounding. An appropriately chosen formal language has the additional benefits of removing any associated ambiguity, inconsistency and incompleteness. Formalisation also provides an opportunity for reasoning (i.e. the deriving of conclusions) and validation. Utilisation of a formal foundation reduces the complexity associated with implementation, increasing automated support.

- **Criterion #2: The service description language must be flexible enough to support the description needs of service providers.** The service representation language should have adequate *expressive* power to represent all non-functional service properties. Expressiveness allows us to model non-functional properties from many services across various domains. An expressive solution should allow service providers to describe their services in a manner that is easy. Expressiveness as a criterion requires a balance. On one hand we seek to provide the ability to describe the non-functional properties of services in a number of ways (e.g. in describing temporal intervals) versus the ease of discovering services based on non-functional properties that have been expressed differently. The expressiveness of the language must capture the static and/or dynamic aspects of the service. The language must also be capable of expressing the relationships between service properties.

- **Criterion #3: The service description language must support the (semi-)automated interactions between service catalogues, service providers and service requestors.** To achieve this the service representation language should be *machine-processible*. This criterion seeks to reduce the human involvement in the service discovery and comparison tasks. Demonstrating this criterion is a precursor to showing the comprehensiveness of the language. To achieve a level of machine-processing the service description language must be available in a concrete form. In this manner it can be used in interactions between requestors, providers, catalogues and brokers.

- **Criterion #4: The service description language must be able to be understood by human beings. In addition, the language should aid the description of the service.** The service representation language should be *comprehensible*. Human beings should be able to understand the language. The ability for human beings to understand the language may be useful in the context of visual tools that are used for composing, selecting and assembling services. Services that are assembled and provisioned electronically are catered for in the previous criterion. Comprehensibility should aid the service provider in developing a description of their service, reducing the time required to specify its description.

- **Criterion #5: The service description language will not be tied to a particular implementation technology.** The service description language should be *conceptual*. This criterion introduces a separation of concerns between the concepts in the language and its implementation. A sufficiently conceptual language will seek to minimise the number of irrelevant concepts. A solution that is conceptual will not attempt to dictate the implementation.

- **Criterion #6: The service description language must capture and be capable of capturing all relevant service related concepts.** The service description language should be *suitable*. Suitability ensures that only relevant domain concepts are mapped into the service representation language. Conversely, the language must also be capable of providing all the concepts of a domain.

- **Criterion #7: The service description language must not inhibit extension to itself by the service catalogue or the service provider. This includes inhibiting extensions into domain specifics.** The service description language should be *extensible*. Extensibility will ensure that domain specific non-functional properties will be able to be included within the language. This criterion also applies to the ability for the non-functional properties of services to evolve over time.

It is important to note that the only true representation of a service is the service itself. Developing this language is an attempt at defining a "surrogate" for the service [30]. With this in mind, it is fundamentally important that the representation

be as precise as possible. This will improve the accuracy of the reasoning (an effect of the formal criterion) that can be conducted with the representation.

1.4 Our approach

We have previously claimed that non-functional properties were an essential component of the characterisation of any service [33, 68]. To support this claim we undertook a significant analysis of services from numerous domains. In the process we have extracted hundreds of non-functional properties that have been subjected to a set of questions that we developed. These questions helped establish whether a property should be included or not within our work. These questions are as follows:

- Does the example being used to test the model present itself in numerous services across multiple domains? We cannot expect to represent every service. An approach that tries to support such a range of non-functional properties will inevitably become cumbersome. How much specification effort is required? Do the benefits outweigh the cost/effort involved? An example of this test is the use of entities within our model that relate to conditions and procedures. Whilst capturing useful information the effort involved in capturing their domain independent properties is significant.

- Have we stepped into a specific domain by including a certain non-functional property within our model(s)? If so, the non-functional property is excluded from use within our models. For example, the writing of personal bank cheques in some countries results in the originator of the cheque being charged a government levied charge or tax. We do not model the specific charge name but are still able to capture the additional charges that may be imposed on the use of the service. We attempt to remain domain independent in our approach.

- Does the inclusion of an abstraction of a concept result in the loss of semantically useful information? In looking at the Universe of Discourse we sometimes attempt to utilise an abstraction of a concept. However sometimes this results in the loss of semantically useful information (i.e. Does it limit the conceptual queries that we can apply to a model)? If the querying of the model is limited by providing an abstraction for a concept then we do not pursue that abstraction. This criterion typically applies when determining subtype hierarchies. Common properties are attached to the supertype with only the relevant properties being attached to the subtype. If by including the property as a generically named role in the supertype then we have made the querying of the model more difficult. This sometimes comes at the cost of not having a pure subtype hierarchy.

- Do we require some context from the requestor to be able to complete our description of the service? We make the assumption that a service provider is describing the service prior to its advertisement then the service provider will be unaware of the context of the service requestor. For example, we cannot

state the price of a service with specific discounts included as discounts require knowledge about the service requestor (e.g. age). We therefore state price and discounts separately, expecting that the service catalogue will take a more active role in the discovery process by using information that it may know about the service requestor and applying it to the discovery of information within the catalogue. With that in mind, we ask if the example being tested requires knowledge that is not available to the service provider at the time of description?

- Do the non-functional properties of a particular model still hold (roughly) under the application of composition and substitution? A primary area of interest for future case studies are the areas of service substitution and composition. Whilst we provide a more detailed description of substitution and composition in chapter 2, the increased level of service description that we advocate enables more detailed comparison of services. The current, manually-intensive situation requires significant effort to compare services. We believe that it will be possible to substitute services within a composition, and generate new service compositions more easily.

From our analysis we compiled the non-functional properties into a series of eighty (80) conceptual models that are present herein. We have categorised the models according to availability (both temporal and locative), payment, price, discounts, obligations, rights, penalties, trust, security, and quality. This content has been published on the Web as a set of navigable models [69].

Our motivation is to provide a theoretical basis for automated service discovery, comparison, selection, and substitution. We prefer not to distinguish between conventional, electronic and web services. Historically, services have always been distinguished according to some criteria of a service requestor (such as price, payment alternatives, availability, security etc). We are motivated to ensure that the criteria used to evaluate conventional services are also available for web services. We are motivated to ensure that service providers can publish descriptions in a manner suitable to their needs. For example, where a service provider wants to outline when a service is available, rather than specifying exact dates they may prefer to outline recurring temporal intervals. We envisage that service catalogues will evolve to a point where localisation of information is primarily for service requestors, and where the context of the service requestor is used to facilitate this localisation. Our approach seeks to offer a domain-independent method for describing the non-functional properties of conventional, electronic and web services.

Our models offer the interesting capability of recursion. In describing the non-functional properties of services there are numerous instances where references to other services or providers are used. For example, a service provider can outline the warranty offered for their service but state that the warranty is fulfilled by a separate provider. Queries over our models then allow us to not only filter based on the criteria of the service of interest, but also over the non-functional properties of the associated services/providers.

1.5 Scope

This thesis focuses on providing a taxonomy for the non-functional properties of services that is capable of representing both conventional and electronic services. By definition we believe that a taxonomy for services is therefore *domain independent*. With the notion of accurate service representation in mind, we guided the work to develop our taxonomy by following a number of fundamental principles. Figure 1.1 provides an overview of our research boundaries. The general problem area (i.e. accurate service description) is represented as an oval, whilst the specific problem of description of service properties is depicted as a rectangle. We now present a small discussion on each of the research boundaries.

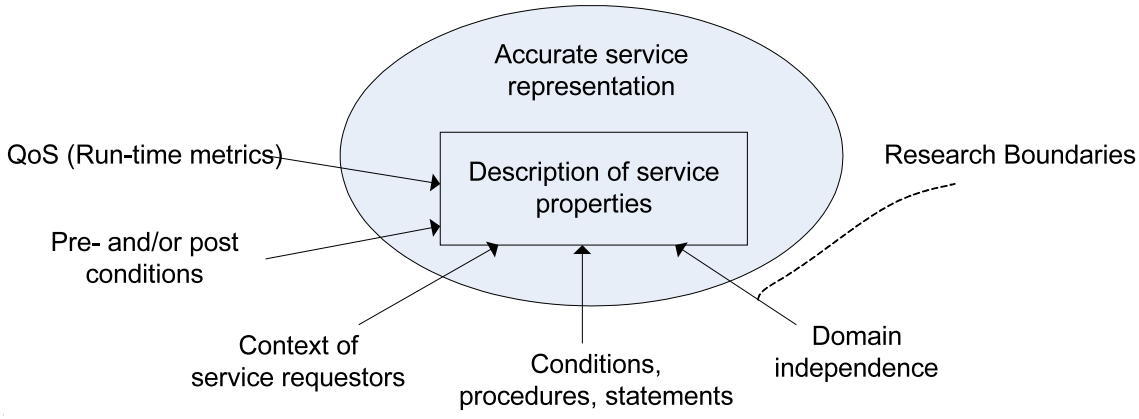


Figure 1.1: Boundaries of research problem

Service requestor context: We specifically consider that a service provider would like to describe contextual information about the service requestor that impacts the description to be presented (e.g. age based discounts). We do however realise that this affects the description of the service. The provider doesn't want to present all permutations of the description based on the distinct contexts that are considered (e.g. publish a description that is at the full price, and publish a description that is at a reduced price for pensioners). It is important to highlight that this view is primarily because the service provider is undertaking the description process prior to the publishing of a service advertisement (perhaps to a service catalogue). This means that the description for a service must be sufficiently well defined to cater for the different types of service requestors, but it is not until the requestor undertakes discovery that the exact nature of the service is known (e.g. we don't know the age of the requestor to determine if they should get a senior citizen discount).

Pre- and post conditions: Pre- and post conditions reflect an assertion that we expect to be true prior to and after a service call respectively. An example of a pre- and post condition is as follows. A university may choose to offer a service

that allows student to unenrol from a particular subject. A pre-condition is that the student is enrolled in the subject. A post condition is that the student is no longer enrolled in the subject. It is our belief that the inclusion of pre- and post conditions would constitute a functional intrusion on our non-functionally focused work. This is not to say that pre- and post conditions are not important for the description of services, they most certainly are.

Conditions: We use the term *condition* generically to refer to the notion of terms and conditions that are common with legal documents (e.g. loan applications). Within our work we do not attempt to define either the syntactic structure or the meaning of conditions. We feel that this constitutes an intrusion into rule languages. Attached to the service may be a list of terms and conditions. These conditions are formalised in a contract and govern the responsibilities of all parties involved in the service request and provision. Contracts are considered binding agreements between parties [19]. Familiar examples include the terms and conditions associated or expressed with items such as credit card applications, loan applications, tickets for transportation or entertainment, and policies (e.g. insurance). We assert that these contracts are representations of the promises of each party. Both parties must be agreeable with respect to the contract before it is invoked. An implementation of the infrastructure required for electronic contracts is outlined in [46]. In the case of a warranty, contracts may prove to extend the life of a service beyond the initial transfer of value. We refer to conditions using Uniform Resource Identifiers [9]. Where we describe a non-functional property that is of a condition type then we provide the ability to refer to multiple conditions. This allows the service provider to determine the appropriate level of granularity for the condition(s) being expressed.

Procedures: Within our work we do not attempt to define either the syntactic structure or the meaning of procedures. We feel that this constitutes an intrusion into workflow and business process management. Procedures are common in companies and we are interested in providing references to externally advertised procedures (e.g. for refunding money) rather than internal procedures such as personnel performance management or petty cash handling. In a computing context procedures are considered to be a sequence of steps. We take the same view for business activities. The procedure defines the necessary detail to show how to achieve the business activity. In the same way as conditions we refer to procedures using URIs.

Statements: Within our work we do not attempt to define either the syntactic structure or the meaning of statements. An example of a statement is a corporate mission statement. We feel that this constitutes an intrusion into knowledge management. A minimal number of references within our work are made to statements. We feel that these vary dramatically and any attempt to define the structure and semantics would always be incomplete. In the same way as conditions and procedures, we refer to statements using URIs.

Quality of service: Numerous notions of quality of service are in fact a discussion surrounding metrics that can be captured during service provision [56, 57, 19, 93]. Our intent with non-functional properties relating to quality is to identify standards, benchmarks and rating schemes that the service attempts to comply with or compares themself with. The comparison allows a service provider to provide an indication of the reliability, responsiveness and/or efficiency of their service. We do not attempt to provide a runtime perspective of what is tracked with respect to quality of service. This is due to the assumption that description of a service occurs prior to publishing within a service catalogue.

1.5.1 Onus of description

Our view of service description is that there is a significant effort required by the service provider. In some instances we believe that the service catalogue may provide relief from the onus of descriptive effort. Particular examples include: well-known standards, common routes and/or regions, common addresses, public holidays (specific to certain countries) etc. The catalogue provider could ensure that this instance data is available through some form of user interface at the time of service description. An alternative view is that this type of information could be generally agreed upon and published to a well-known location. This would make it available independently of the service catalogue being published to and would ensure a level of compatibility between service catalogues.

1.5.2 Object-role modelling

We make extensive use of the Object-Role Modelling (ORM) [48] conceptual modelling technique within this thesis. This technique is sometimes referred to as NIAM - Natural Language Information Analysis Model. Our reasons for using ORM are numerous. Firstly, ORM is a rich and expressive modelling technique. The models are capable of being verbalised and validated by a domain expert. Secondly, due to ORM's fact oriented view of a domain, it is more stable to changes that occur within an application domain. This is distinct from attribute oriented modelling techniques such as ER and UML. Thirdly, ORM is capable of capturing more business rules in diagrammatic form than both ER and UML. Being a logical modelling technique, ER modelling is removed from natural language. An overview of the ORM notation is provided in Appendix 8.4.

Within this thesis we are using ORM to undertake *ontological modelling*. As outlined in [52], conceptual models provide a design-time aspect but we also make use of it within a run-time context. This run-time context is presented in chapter 7. According to Jarrar, the "relative generic knowledge" captured within an ontology sets it apart from the application dependent needs of a conceptual model. Having said that we are undertaking ontological modelling, the question may well arise as to why we phrased our research question using the term taxonomy? We set out to develop a detailed taxonomy rather than an ontology that included derivation rules. It is for this reason that we define our work primarily as taxonomy.

1.6 Alternative approaches

This section presents alternative approaches to our work. We categorise the approaches according to web service related standards, semantic web initiatives, collaborations between industry partners, proprietary service catalogues, standards that leverage semantics and some miscellaneous approaches.

1.6.1 Core web service standards

Prevalent Web services standards [41, 23, 13, 47, 104] offer a purely functional focus, with their combined intention to be the exposure of coarse grained business functions for consumption by service requestors. In reality they act as a common formatting idiom [45]. The HyperText Transfer Protocol (HTTP) provides the communication protocol that is used on the World Wide Web to transmit requests and receive responses. The Extensible Markup Language standard (or XML) provides an approach for describing the structure of data within a markup language. Simple Object Access Protocol (SOAP) is an XML based protocol that offers an envelope that describes what is contained within a message and how to go about processing it, as well as an approach to representing a remote procedure call. The Web Service Description Language is an XML format for describing services and their operations that are exposed as a network endpoint. Messages are used to describe the input and output of operations.

Unfortunately, Web services standards do not adequately assist with the description of conventional services. There is no semantics provided for the messages being exchanged between service parties. It appears that Web services are treated as "computational entities" rather than also viewing them as an electronic request mechanisms for conventional services. For example, a web service might be used to expose a booking service for a chiropractor. In this instance, the standards have aided with the request perspective of the service, not with the provisioning of the service.

Universal Description, Discovery and Integration (UDDI) provides a method of both publishing and discovering service descriptions [104]. It acts as a registry of service descriptions that is segmented into three areas - white pages (providing contact details), yellow pages (providing industrial categorisation) and green pages (providing technical information). UDDI captures items such as the business entity, the business service, details about how to invoke the service (referred to as a bindingTemplate) and meta-data relating to a specification (e.g. name, publisher, URL). UDDI provides sufficient capability to publish a rich service description but as with the other Web services standards it lacks semantics. An extension to UDDI, termed UDDIe, offered the notion of blue pages to record user defined properties [92]. This would allow for specific service provider properties, such as non-functional properties, to be captured with respect to their service.

Representational State Transfer Representational State Transfer (REST) is an architectural style (not a standard) [42]. Quite simply, clients access a URL

using the HTTP protocol in a stateless manner. A representation of a resource is returned from the server. Since the client has received the representation it is considered to have undergone a state change. Representations typically take the form of XML or HTML. Resources are identified in a "RESTful" system using URIs [9]. RESTful applications are built on standards that have made the World Wide Web a success, so standards such as SOAP, WSDL and UDDI are considered unnecessary. The research presented in Fielding's thesis is capable of being represented with applications that support the common Web services standards, or that exhibit RESTful characteristics. Neither approach provides the semantics necessary to undertake service discovery, selection or comparison. Hence our work is considered complementary to these pieces of work.

1.6.2 Semantic web initiatives

Two main semantic web initiatives are relevant to our work, the Web Services Modeling Ontology and OWL-S. Each of these initiatives attempt to semantically enrich Web content through the use of a service description language and ontologies. The subsequent effect is the ability to perform automated discovery, selection, composition, substitution and invocation.

Web Service Modeling Ontology (WSMO) WSMO [82, 15] provides a conceptual model for describing Semantic Web services, the combination of Semantic Web technology and Web services technology. WSMO, based on the Web Service Modeling Framework [40], provides for the description of four elements - ontologies, goals, Web services and mediators.

For all these elements, WSMO allows for the attaching of non-functional properties. The non-functional properties primarily consisted of the Dublin Core [114] metadata element set plus an additional "version" property. These base properties were then extended to include some web service properties. Unfortunately these were more indicative of the categories of non-functional properties (i.e. financial, language, performance, reliability, trust, robustness) than a specific non-functional property set. Subsequently, the non-functional properties within WSMO have been almost entirely based on research contained within this thesis [102]. This work has resulted in the formalisation of our research in the Web Services Modeling Language (WSML) [16].

OWL web service ontology The OWL web service ontology (OWL-S) [73] has a similar purpose to the WSMO semantic web initiative. It attempts to provide a framework for describing both the capability and properties of Web services. Formerly called the Darpa Agent Markup Language web service ontology (or DAML-S), OWL-S refers to service descriptions as having three different types of information - a profile, a process model, and a grounding. The profile provides an outline of what the service does, the process model assists a service requestor by providing an understanding of how the service works, whilst the grounding outlines the details of how to access the service.

The area of OWL-S of interest to us is the service profile. Earlier versions of DAML-S [5] committed to some basic non-functional properties within the profile (e.g. geographicRadius, degreeOfQuality, qualityGuarantee, communicationThru), however these were subsequently deprecated. The underlying view from the OWL-S Coalition is that the non-functional properties of services are domain specific. They provide the mechanism to capture these properties but prefer not to specify significantly more than the service name, a text description and contact details.

Whilst we don't subscribe to the domain specific view of non-functional properties that is held by the OWL-S Coalition we know that the work contained within this thesis has been converted to the Web Ontology Language (OWL) and could theoretically be used within OWL-S profiles. We prefer to think of OWL-S' view of non-functional properties as "placeholders".

1.6.3 Industry collaborations

This section provides an overview of alternative approaches that are being driven through industry initiatives. Our general comment about the relevance of these works to ours is that they are particularly focused on runtime execution. Those parts that are focused on discovery are more with respect to the business process being executed. The properties available within these works can be considered Electronic Data Interchange, hence they are pursuing the valid cause of a lower cost of business through better efficiency/productivity. These standards do have non-functional properties represented within. We feel however that our research presents richer semantics for temporal and locative availability, rights, obligations and payment.

ebXML At its core, the ebXML initiative provides an ability to describe a business process, and an associated information model [36]. Once described, business processes can be shared and re-used by other trading partners. Trading partners publish into the ebXML registry the business processes that they support, the business interfaces, the messages that must be exchanged as part of the interaction and technical configuration details (e.g. security). Other trading partners use the ebXML registry for discovery purposes.

As a means of describing their business practices, it is recommended that organisations use the UN/CEFACT Modeling Methodology (UMM) [107]. UMM provides two perspectives of business processes - the Business Operational View (BOV) and the Functional Service View (FSV). The intent of the BOV is to: (a) provide the semantic underpinning for interactions that occur within ebXML; and (b) to define the architecture for business interactions (e.g. operational conventions, agreements, arrangements etc). It is this first intent surrounding semantic underpinning that is of relevance to our research. Of interest within the BOV is the aspect of core components. Core components are intended to capture business concepts and their interrelationships. We believe that it is this ability to capture core components that allows for our work to be used within ebXML. This is evidenced by the discussion that follows surrounding the Universal Business Language.

Universal Business Language The first version of the OASIS Universal Business Language (UBL) was intended to provide XML document support for the business functions that exist from ordering through to invoicing [67]. This included the order, order responses (simple and detailed), order amendment, order cancellation, despatch advice, receipt advice and invoicing. Subsequent versions of UBL have expanded that original set of XML documents to include support for an extended procurement process, support for buyer to buyer transactions, support for marketplaces, better European Union support and enhanced localisation.

UBL works in conjunction with the ebXML standard as it can be used as the message payload for ebXML messages. ebXML has defined business terms that are independent of the documents within which the are used. Examples of terms are Order, Quotation, ReceiptAdvice, Reminder, Statement and CreditNote. These business terms are referred to as ebXML Core Components and are presented in the ebXML Core Components Technical Specification (CCTS) [108, 106]. UBL is considered to be an implementation of the ebXML CCTS.

RosettaNet Similar to the business processes available within ebXML, Rosettanet defines Partner Interface Processes (or PIPs) [85]. PIPs have been defined in such areas as product information, order management, inventory management, marketing information and support. RosettaNet provides dictionaries as a mechanism for describing common properties that may be used within PIPs. Two types of dictionaries are available - Technical Dictionaries define products, whereas Business Dictionaries describe the properties used within business processes.

Business Dictionaries within RosettaNet closely align with our work. Concepts such as account, customer, payment (debit/credit), discounts, delivery, product, remittance are available [84]. We believe that our research is more targeted with respect to services, whereas the business dictionaries appear to be more focussed on product orders.

1.6.4 Proprietary service catalogues

We can categorise proprietary service catalogues into those that are currently used for conventional services (e.g. Yellow Pages) and those that are public marketplaces. Sites like Yellow Pages offer basic provider and service related information. For example, provider name, web site, email address, operating hours, payment mechanisms accepted, number of employees, and date established [121]. These properties are intended to provide sufficient information to filter candidate service providers. We do acknowledge that some level of temporal and locative availability, proprietary classification and trust are attempted to be provided to service requestors.

Services made available through public registries or marketplaces such as [91] tend to provide a range of unrelated services. For example, currency conversion, distance calculator, weather report, and package tracking. Each service provider utilises a WSDL file to functionally describe their service. No common service terminology appears to be used by service providers.

1.6.5 Standards leveraging semantics

Four other standards that are more peripheral to the core web services standards are the Web Service Level Agreement, WSDL-S, WS-Policy and the Web Services Inspection Language. These approaches all leverage non-functional property semantics that are external to the standard. This allows the standard to be utilised with various non-functional property semantics without committing to a particular non-functional property semantics work.

Web Service Level Agreement Web Service Level Agreement (WSLA) allows service providers to make assertions about the provision of Web services [59]. It is both a runtime architecture (that includes monitoring) and a language. Various monitoring services related to the negotiation and authoring of a service level agreement (SLA) - measuring metrics, checking SLA parameter thresholds and taking corrective actions. SLA's within WSLA primarily consist of parties, service definitions (that capture metrics), and obligations (either a service level or action guarantee). WSLA should be viewed as complementary to service description languages. It provides a useful mechanism for expressing the level of service that a requestor can expect, and a means of measuring and managing SLA's.

WSDL-S WSDL-S is an attempt to address the issue of the lack of semantics within Web services related standards [1]. It builds on existing standards, and extends WSDL 2.0 [22] to support annotation of input and output messages, annotation of complex types, and the specification of pre-conditions and effects. Lesser support for extension of WSDL 1.0 [23] is also available. WSDL-S is primarily interested in the semantic annotation of abstract aspects of WSDL. This limits it to the interface, message and operation elements contained within the language. Implementation aspects of the WSDL such as the binding, endpoint and service elements are ignored.

WSDL-S is viewed favourably by tool builders as it represents an incremental change to existing tools. This is unlike the changes being proposed by semantic web initiatives such as WSMO and OWL-S. WSDL-S represents a simple, yet effective approach for the semantic annotation of WSDL. As such it is complementary to our research. To further support this claim we present a substantiation of our research in Chapter 7 that involves the semantic annotation of WSDL using WSDL-S.

WS-Policy The Web Services Policy Framework provides a generic mechanism for describing the "capabilities, requirements, and general characteristics of entities in an XML Web services-based system" [7]. The framework does not provide a description of how the policies are discovered or attached to a Web service. An associated standard, WS-Policy Attachment caters for attachment to any XML document [8]. It has also been designed to work with other Web services related standards such as UDDI [104] and WSDL [22]. Policies are expressed as *policy assertions*. Each assertion is considered to apply to *policy subjects* and specifies a requirement of that subject.

Our work can be used to provide policies surrounding non-functional properties within the WS-Policy framework. Whilst we have chosen to substantiate our work (see Chapter 7) we have not used WS-Policy as one of those substantiations. We consider it a complementary standard for our work, and yet another place for it to be used.

Web Services Inspection Language The Web Services Inspection Language (WSIL) has two primary functions: to aggregate existing service descriptions (using an XML format) and to act as a distributed service discovery mechanism [14]. WSIL does not attempt to act as a service description language. It is capable of working with a variety of service description formats. WSIL documents can be utilised in one of two conventions: located in a static position of the web site, or as a META tag within a HTML document. WSIL documents contain one or more service elements. Service elements contain one or more references to different types of service descriptions (for the same service). WSIL documents may also link to other WSIL documents. Link elements contain a reference to only one type of service description. There is also WSIL extensibility for WSDL and UDDI. Due to WSIL's support for various service description formats, we consider it to be complementary to our work.

1.6.6 Miscellaneous

Process NFL A language for describing non-functional properties called Process NFL is outlined in [83]. The language is primarily intended to capture the non-functional requirements of software during the software development process. Three concepts are at the core of the language. Non-functional properties are expressed as NF-Attributes, these can be constrained using NF-Properties. A final concept (NF-Actions) relates to the design and hardware considerations. This highlights the language' specific usage during software development. Positive aspects of the language are that it offers the ability to specify priorities and constraints, and the ability to correlate different non-functional properties. To its detriment, Process NFL does not have a formal semantics and this is of serious concern with respect to our first criterion for a solution presented in section 1.3. Properties presented in subsequent chapters of this thesis could be put into the Process NFL language but its lack of formal semantics means that this would be unproductive.

1.7 Contribution of the research

We believe that the primary contribution of this research is a domain independent taxonomy capable of representing the non-functional properties of services for both conventional and electronic services. The concrete representation of this taxonomy (presented in Chapter 7) provides the ability to communicate non-functional properties of services as part of a service description, increasing the efficiency of the service discovery processes. The reasoning that can then be performed over service descriptions reduces the candidate set of services discovered by a requestor. This reduction

of candidate services acts to streamline the discovery process. Reasoning can also be performed as part of service comparison and substitution. More generally, the research results in a better understanding of the complex nature of services.

1.7.1 Publications resulting from this research

A journal publication resulting from this research is as follows:

- J. O'Sullivan, D. Edmond, and A. H. M. ter Hofstede. *Whats in a service?: Towards accurate description of non-functional service properties.* Distributed and Parallel Databases Journal - Special Issue on E-Services, 12(2-3):117133, 2002.

Conference and workshop publications resulting from this research are as follows:

- M. Dumas, J. O'Sullivan, M. Heravizadeh, D. Edmond, and A. H. M. ter Hofstede. *Towards a Semantic Framework for Service Description.* In R. Meersman, K. Aberer, and T. Dillon, editors, Proceedings of the 9th International Federation for Information Processing (IFIP) Conference on Database Semantics - Semantic Issues in e-Commerce Systems, volume 239 of IFIP International Federation for Information Processing, pages 277291, Hong Kong, China, 2001. Kluwer Academic Publishers.

- J. O'Sullivan, D. Edmond, and A. H. M. ter Hofstede. *The Price Of Services.* In F. Casati, P. Traverso, and B. Benatallah, editors, Proceedings of the The Third International Conference on Service-Oriented Computing (ICSOC05), pages 564569, Amsterdam, The Netherlands, 2005. Springer Verlag.

- J. O'Sullivan, D. Edmond, and A. H. M. ter Hofstede. *Two main challenges in service description: Web service tunnel vision and Semantic myopia.* In W3C Workshop on Frameworks for Semantics in Web Services, Innsbruck, Austria, 2005.

- J. O'Sullivan. Towards a Precise Understanding of Service Properties. In P. Albers and D. Lopes, editors, Proceedings of the 1st ICEIS Doctoral Consortium (DCEIS-2003), pages 3033, Angers, France, 2003. ICEIS Press.

- J. O'Sullivan, and D. Edmond. *RFT: a useful model for dynamic web processes.* International Workshop in Dynamic Web Processes (DWP 2005), Amsterdam, The Netherlands, 2005.

- J. O'Sullivan and D. Edmond. When and where is a service? Investigating temporal and locative service properties. In Proceedings of the Symposium on Applications and the Internet Workshops (SAINT 2003 Workshops) - Service Oriented Computing:Models, Architectures and Applications Workshop, pages 9094, Orlando, FL, USA, 2003. IEEE Computer Society.

Other publications resulting from this research are as follows:

- J. O'Sullivan, D. Edmond, and A. H. M. ter Hofstede. Service Description: A survey of the general nature of services. Technical Report# FIT-TR-2003-01, Queensland University of Technology, Brisbane, January 2003.

- J. O'Sullivan, D. Edmond, and A. H. M. ter Hofstede. Formal description of non-functional service properties. Technical Report# FIT-TR- 2005-01, Queensland University of Technology, Brisbane, February 2005.

1.7.2 Uptake by standards

The contents of this research has been almost entirely utilised within the Web Services Modeling Ontology (WSMO) [82, 15], a standard developed by the Digital Enterprise Research Institute. This semantic web initiative provides for the attachment of non-functional properties to four key elements - ontologies, goals, Web services and mediators. DERI's original attempts to describe non-functional properties included the Dublin Core [114] metadata element set (which are all optional), plus a "version" property. Other non-functional property categories (e.g. financial) were outlined but the details were never expanded upon. Our work completes the detail within each of these categories.

The inclusion of our research within WSMO has provided a key differentiator over other semantic web initatives such as OWL-S. It also acts as a validation and subsequently we expect to see numerous WSMO semantic web applications making use of these non-functional properties.

Summary

Whilst we acknowledge the importance of service functionality, this thesis is primarily concerned with the non-functional properties of services. A service is *not* a function. It is a function performed on your behalf at a cost. And the cost is not just some monetary price; it is a whole collection of limitations. This thesis is all about these. We consider the non-functional properties to be constraints over the functionality of the service [24]. We believe that a service description is only complete once the non-functional aspects are also expressed.

The non-functional properties of services introduce complexity to the description of services but their inclusion is crucial to the automation of service discovery, comparison and substitution. We have stated in this chapter our belief that two main challenges confront the future of service description - overcoming web service tunnel vision and overcoming semantic myopia. That is, choosing to ignore both the rich history of conventional services, and the non-functional properties of services (perhaps through deferring to domain specific ontologies, or by a continued functional focus). We also believe that accurate service representation promises to reduce the gap between conventional, electronic and web services.

The chapters that follow attempt to define services, and more importantly detail the non-functional properties (i.e. temporal and locative availability, obligations,

rights, payment, price, discounts, penalties, quality, security, and trust) that comprise services. It is our opinion that this work has managed to utilise conventional services as the basis for describing a range of domain independent non-functional properties. This provides an opportunity for expressing the non-functional properties of services using a single technique for conventional, electronic and web services.

Chapter 2

Conceptual Foundation

A proper understanding of the general nature, potential and obligations of electronic services may be achieved by examining existing commercial services in detail. The everyday services that surround us, and the ways in which we engage with them, are the result of social and economic interaction that has taken place over a long period of time. If we attempt to provide electronic services, and do not take this history into account, then we will fail. Any attempt to provide automated electronic services that ignores this history will deny consumers the opportunity to negotiate and refine, over a large range of issues, the specific details of the actual service to be provided. To succeed, we require a rich and accurate means of representing services. An essential ingredient of service representation is capturing the non-functional properties of services. These include the methods of pricing and payment, constraints on temporal and locative availability, obligations, discounts, penalties, quality, security, trust and the rights attached to a service. Not only are comprehensive descriptions essential for useful service discovery, they are also integral to service management, enabling service negotiation, composition and substitution.

2.1 What is a service?

Many definitions for services are based on technology. Some definitions of *electronic* services (or e-services) use the Internet and/or workflow as a conduit to new revenue or task completion [78, 87, 113]. A *web* service has been described as an aggregation of functionality published for use [55]. This is similar to the virtual business processes that define company-level interactions [56]. Other definitions offer a view of services as an abstraction of a business process [20, 90, 53]. We assert that e-services exhibit minimal constraints on the time and location of *request*. There may, however, be a delay between the request and the *provision* of a service. Such a delay may arise because of resource constraints or because of human intervention required in the performance of the service. We believe that an e-service is characterised by its ability to be automatically *summoned* anywhere, anytime.

Surrounding these definitions are three immutable features of services. Firstly, that services are actions performed by an entity on behalf of another to the benefit typically of both parties. Secondly, services are an asset [78]. They have an inherent

value that is transferred from the provider to the recipient. Finally, services can be contained within other services (e.g. a warranty) [123]. We refer to this relationship by describing some services as *sub-services*.

2.2 Service interactions

Service interactions involve three primary participants: a service provider, a service requestor and optionally, a service catalogue (or registry) [20, 21, 31, 55]. A more detailed view has been offered in [35]. A fourth participant, the service broker, is less frequently present in service interactions. We consider service brokers (e.g. an insurance broker) to be entities that offer services from multiple providers to a requestor. Service brokers attempt to add value to the service discovery process, sometimes protecting the identity of the service providers for their own benefit. The service broker can be considered to be a form of anonymized service catalogue. We identify four main interactions between these participants: discovery, negotiation, invocation, and provision. We discuss additional aspects of the life cycle in section 2.4.

The recognition of a need, by a service requestor, triggers the search for suitable service providers. Requestors with little or no knowledge of appropriate providers normally interact with service catalogues. Catalogues are themselves normally well-known services that provide limited comparison between the needs of a requestor and the advertised service descriptions they maintain. The majority of the comparison task is performed by service requestors. This is often due to the varied terminology contained within service descriptions. The names of candidate providers (and some other basic properties), if available, are returned to the requestor. Requestors who are aware of the appropriate provider(s) may bypass the provider search with a catalogue and directly approach the provider. To be able to *discover* a service requires that it be defined, somehow, and that this definition be published.

Requestors contact candidate providers and undertake a series of interactions that refine their knowledge about both the functionality and the non-functional properties (e.g. availability and quality) of each potential service. Generally, this *negotiation* results in a "service contract" that outlines the obligations of each party, and that may or may not be enacted. Requestors have the opportunity to refine their knowledge with respect to multiple candidate providers before making the decision to invoke a particular service.

Invocation is the term we use to identify the process which begins with the formation of a binding agreement between requestor and provider. This is essentially a *call* for the provision of a service. Invocation of the service contract also triggers the provision and/or production of the service by the provider. Services can be invoked using different forms of binding requests. These include electronic requests (e.g. REST style URI request or form-based web page), verbal requests (e.g. buying from a shop), written requests (e.g. purchase order or letter), manual actions (e.g. turning on the TV) and sensor-based requests (e.g. automated door).

We refer to the delivery and consumption of a service as its *provision*, which

normally results in the fulfilment of each party's obligations. Some services are delivered at a location and time distinct from the location and time of the invocation. For example, when booking a trip you might walk to or phone your travel agent. The trip is booked (i.e. invoked) and the service will be rendered at the airport, at the nominated date and time. A specific instance of service delivery can occur within the context of an existing service. This is evident when you catch a bus that is moving between points A and Z. You request the service (i.e. hail and get on the bus) at point D and your consumption of the service ends at point G (i.e. you get off). In the example provided it is interesting to note that the service provider (i.e. the bus company) may offer the service yet it is not consumed (i.e. the bus is driven from A to Z and nobody gets on). This is the characteristic of perishability described in [123]. Consumption may involve the suspension and resumption of the service.

2.3 Functional properties

The description of the behaviour or capability of a service is in essence a description of what the service can do. Service providers should be capable of describing the behaviour that the service provides. We believe that there is a one-to-one mapping between a service's capability description and its non-functional description. We discuss this relationship in more depth at the end of this chapter. In addition, the description of the service capability must include any pre-conditions and post-conditions, the inputs that a service expects in order to provide the capability, the resulting output that the service provides.

Oaks [66] provides a conceptual model for capability descriptions that supports their stated requirements for a capability language. The requirements are supported in various ways. Action verbs are used to declare the action that the service performs. The notion of *signatures* is outlined for the definition of inputs, pre-conditions and effects. These signatures allow a particular capability to have more than one input mechanism. Oaks links a signature to a particular expression within a rule language to achieve their declaration. Definitions of terms are available through *ontological sources* (e.g. a dictionary) to place terms within a particular context. An ability to classify services according to a domain is provided through reference to an ontological source. To assist with exact or partial matches to the capability description an ability to define synonyms is provided. Different verbs can be assigned to ensure greater ability to match descriptions.

In providing this level of detail about the behavioural or functional aspects of a service, it highlights the level of remaining information that is required to make decisions with respect to a service. This information can be sufficed by the inclusion of the non-functional properties that relate to the service. In section 2.4 and 2.5 respectively we outline the use of non-functional properties before discussing at a high level the categories of non-functional properties.

2.4 The use of non-functional properties

Non-functional properties can be used during the numerous operations of services. The service life cycle is controlled, by the service provider, from conception, to decommissioning where a service is no longer to be offered. It typically involves the definition (including advertisement), discovery (including negotiation), invocation, and decommissioning. All these aspects form part of the general evolution of a service.

Once a service has been defined, one or more descriptions can be generated. These descriptions, sometimes referred to as advertisements or offers, are normally published within a catalogue. Matchmaking is conducted by catalogues using the search criteria provided by a requestor and the descriptions published by service providers. Currently service descriptions are primarily static and have insufficient detail to allow decision making to occur at the service catalogue, or without contacting the service provider. Services, and consequently their descriptions, may require modification as a result of interactions with service requestors, other service providers or their surrounding environment.

2.4.1 Definition

After service conception, a service provider undertakes to describe the service with a view to publishing it, normally within a service catalogue(s). Service catalogues typically utilise proprietary terminology requiring the service provider to undertake description in different ways for each catalogue that will be used. It is within this operation and the operation of discovery that non-functional properties of services are of primary interest. Definition provides an opportunity for a service provider to detail sufficient information about their service that it enables the service requestor to reason over the service description. This reasoning (at the catalogue) reduces the possibility of having to approach each service provider to gather sufficient information to make an informed decision. Less service provider interactions produces efficiencies over the service interactions that currently occur in a manual way. We outline the categories of non-functional properties that are useful to definition and description in section 2.5.

2.4.2 Discovery

As mentioned in section 2.2, we consider discovery to be the process of finding candidate service providers. This does not include the refinement of the requestor's understanding of the service which we consider to occur during negotiation with the service provider. Service catalogues (e.g. YellowPages) currently maintain lists of service providers categorised according to proprietary classification schemes. Non-functional properties are largely restricted to the temporal (e.g. 24x7) and locative (e.g. an address or telephone number) availability, a service name and service category. Temporal and locative availability for the request and provision, quality of service, rights of the requestor over the service, detailed payment and price infor-

mation, security and trust are not widely supported by catalogues for requestors. Inclusion of these non-functional properties within a published description allows for more detailed refinement of candidate service providers to occur through the service catalogue.

For example, a requestor located in Canada, wishes to discover a service that provides stock quotes from the Hong Kong Stock Exchange. The requestor wants to ensure that the following non-functional properties are addressed by the provider: (1) that the software is developed according to the ISO 9001:2000 quality standard, (2) that the request and delivery occurs via the web, (3) that the settlement model is subscription-based, (4) that the charging style is by a unit of measure (i.e. time) and granularity based (i.e. monthly), (5) that payment can be made in US dollars, (6) that the information is no more than 20 minutes old, (7) that username and password security is required to access the service, (8) that they trust the service provider based on the fact that the service has been offered for more than 5 years, and (9) that they have the right to terminate the service after six months with only 2 weeks notice. For sophisticated service requestors, this type of service discovery is not currently possible. It is hoped that this level of description and matchmaking will reduce the need to contact providers only to discover the requestor's requirements do not match the supplied service.

We consider that the result of the discovery process with the service catalogue can lead to one of a number of results:

- Candidate services that meet search criteria are returned to the service requestor. There may be sufficient information to:
  - Make a decision to invoke a service; or
  - Require more specific information from a candidate service provider before invocation can occur; or
  - Decide not to pursue invocation of any candidate services.
- No candidate services were available.

2.4.3 Substitution

Substitution uses accurate service descriptions to allow rational optimisation of services. Taking two services A and B and combining them sequentially may be easy to conceptualise. Service A may be an electronic news report and service B an electronic weather report. If we try to outsource them then difficulties arise. A may only be offered in the USA and B in Chile. Pretty useless if you live in Australia; and pretty useless too if A is available on weekdays and B only on weekends. If, as virtual service builders, we want to configure such a composite service, then the non-functional properties of contributing services must be examined carefully. This discussion raises the notion of *substitutability* in the context of composition. In software engineering, there are established rules about the substitution of one function by another. These rules are captured in the approach known, not coincidentally, as

programming by contract. There, we may substitute one function F by another G if G has weaker preconditions and stronger post-conditions.

Suppose we have, at some time in the past, composed a configuration that contains A, and we encounter another potential service A'. It seems safe to assume that, if A is only available on weekdays but A' is available seven days a week then, all other things being equal, we can substitute the newer one. Thus we may anticipate a number of substitution guidelines. A' may be substituted for A provided:

A' is cheaper than A
A' is more locatively available than A
A' is more temporally available than A

These rules may be compared with weakening the preconditions; for example, a service that is more geographically available has, essentially, weaker conditions attached to its use. Other properties may be associated with the concept of post-condition. For example, a service with *stronger* consumer rights may always be substituted for one with weaker obligations.

2.4.4 Composition

We consider services that contain other services (or sub-services) to be either an aggregation or a composition. Aggregations combine multiple services and provide access to them in a single location. Telecommunications companies can be considered an example of service aggregators. Services such as call forwarding, call diversion and voicemail, are brought together and offered via the telephone. A composition is a tightly-coupled *integration* of sub-services that results in value not present within the individual services. This added value may be represented in terms of another service property (e.g. reduced price, increased trust). Within a composition, each sub-service is a service in its own right and complex inter-relationships may exist between the sub-services. Service composition should not be confused with functional composition. It has a broader goal that needs to take into account both functional *and* non-functional issues. It may be that we can (functionally) compose a transportation service by articulating land and air transport services. We may equally (non-functionally) compose some hitherto free service with a payment mechanism to form a commercial version of the original service.

Composition is a way of defining a new service. Static or dynamic composition requires an accurate and detailed understanding of the services involved. As a composer of services, discovery and substitution are integral. Discovery provides an opportunity to determine service providers that can be included in a composition, whilst substitution is useful for existing compositions where a sub-service needs to be replaced. Lets look at an example. An entity determines that they would like to compose a new service that provides hotel and car rental bookings. An appropriate hotel reservation service, and a vehicle reservation service must be found. The new service is to exhibit the following non-functional properties (1) it is to provide a single settlement model, (2) it is restricted in locative availability accommodation and car hire in France, (3) it is restricted to service requestors from Australia, (4)

the accommodation is rated as greater than 3 stars, and (5) the vehicles need to be restricted to carrying greater than 4 people. The composing entity needs to discover services that meet the specified criteria. To undertake this in a dynamic manner, sufficient functional and non-functional information must be included with its published description.

2.4.5 Management

Rich repositories of service metadata provide an opportunity for monitoring and controlling the operations that occur on that metadata (e.g. discovery, substitution, composition, provision). Existing service management architectures that support composition include Aurora and DySCo [61, 78]. We suggest that any service management architecture that aims to monitor or control service life cycle operations will need to recognise these operations by means of a rich service description language. However, such a system will need to do more. These systems may be relied on to establish that the behaviour of a service, as delivered, is consistent with the service as specified in a contract is a highly important issue. *Conformance* or *compliance* may have legal consequences. How can it be demonstrated, by examination of a trace or otherwise, that a service was or is being properly delivered?

Additionally, service management repositories offer opportunities for the development of comparative tools that evaluate services "side by side" and that are capable of tracking the evolution of a particular service or type of service. As services evolve, consequently their descriptions should also reflect that metamorphosis. Evolution of a service can be the result of (a) interactions with either requestors or service composers, (b) changes to the environment that surrounds a service, (c) the need to alter the functionality, or (d) impetus from the changing constraints or non-functional properties over the service. Mechanisms that implement non-functional properties (e.g. security, trust and channels) will evolve with standards from the relevant domains. Service evolution is likely to be constrained by the existing commitments that service providers have to delivering a service. The need to administer evolving service descriptions questions the need to include expiry conditions (e.g. temporal constraints) within the description. This provides a mechanism for updating cached descriptions. A similar mechanism is provided in HTML metadata.

2.5 Categories of non-functional properties

We now present a discussion of the categories of *non-functional* properties associated with services. As previously mentioned we consider the non-functional properties to be constraints exhibited over the functionality of the service. The categories of non-functional properties of services include temporal and locative availability, payment, price, obligations, rights, quality, security, trust, penalties and discounts. In the subsections that follow we reveal sufficient detail with respect to each category of non-functional property as to outline the complexity involved with accurately describing it.

The categories presented within this section were initially developed during the writing of [33]. Subsequent research, and the use of service description examples have both changed the non-functional properties that we are interested in, as well as impacting the categories that we use to logically group the non-functional properties. Refinement of the categories also occurred using the criteria that determined whether a property was included in our work (presented in section 1.4). Finally, with the use of many service description examples, we were able to discern the final categories from the relationships between entities within the ORM models. We used groupings of entities within the ORM models to determine logically related entities.

2.5.1 Availability

We consider *availability* to refer to the temporal (i.e. when) and locative (i.e. where) constraints applied to a service. Availability is a complex property of services. For example, there are services that are regularly on the move (e.g. taxis, trains). There are also services where an implicit understanding effects the advertised availability (e.g. when attending the theatre, you need to be in the lobby prior to the start time so that seating can take place.) Thirdly, there are services where there is a suspension and resumption (e.g. memberships).

In representing complex availability information other issues, apart from those outlined, need to be addressed. Often, services quite intentionally provide incomplete temporal and locative information. For example, when you buy an airline ticket you know the airport where the plane departs from. Further refinement occurs at check-in to include a departure gate number, boarding time and a seat number. Availability of a service may be specified with respect to another object (e.g. an emergency phone is available 3 km south of a particular overpass on the freeway). This is also referred to as orientation and is defined using the primary object, a reference object and a frame of reference [27]. Different temporal representations can be used but they assume a "degree of certainty" about the information being represented [4]. Uncertainty increases with a reduced frequency of sampling [76, 88]. How do we know that a bus will arrive according to its timetable? We can assume that it will arrive on time or we can stand at the bus stop and continually check. It is important that uncertainty is communicated to the service requestor. Some services have exclusivity arrangements relating to their availability (e.g. an appointment for the doctor or hire of a conference centre). Location-based services (e.g. where is the nearest hotel to where I am now?) also face the same representational challenges.

For decision-making reasons, service requestors may need to be aware of more than just the availability of service request and provision. To enable accurate scheduling of multiple services, the requestor may be specifically interested in the duration of the service or the approximate completion time. These may be required when performing service discovery, advertising, composition, and when determining service quality.

Provision of a service may utilise broadcast techniques. This is a means of addressing an unknown number of providers (e.g. placing a wanted advertisement in the *classifieds* section of a newspaper) or requestors (e.g. receiving news updates

from a web site). This technique is more commonly referred to as *pushing*. Broadcasting has the unique property that it may not have been explicitly requested (e.g. a television or radio station) and additionally, may have no associated request mechanism.

Temporal and locative representation

Temporal representations need to support various granularities or alternatively represent time as a relationship (e.g. service "X" begins after service "Y"). Common temporal granularities include seconds, minutes, hours, days, weeks, months and years. Approaches for capturing these granularities and their relationships (e.g. finer-than, groups-into) have previously been offered [10, 2]. Temporal database literature has well-defined terms such as chronon (non-decomposed unit of time), time stamp, lifespan, event and interval [54]. Analogous to chronons is the concept of a moment [3]. Each of these concepts offers insight into the expression of granularities for temporal availability. A useful summary of the problems associated with using temporal time stamps such as *now* are outlined in [26]. Another method for representing date and time is the ISO standard 8601:2000 [49], which is intended for use in software to software exchanges.

The artificial intelligence community uses dating schemes, constraint propagation and duration-based schemes for temporal representation [4]. Within the spatio-temporal database community sets of object, location, and time-stamp triplets have been used to represent time evolving spatial objects [100]. Three temporal specification issues are outlined, each of which is applicable to services: (1) data type support for service definition languages; (2) index construction for service catalogues; and (3) query processing for service discovery.

Spatial representations are used to describe topologies, orientation, shape, size and distance [27]. A discussion of spatial models and their classifications (comprehensiveness, structure, theoretical foundation, modelling techniques) is found in [43]. Latitude, longitude and altitude (e.g. for planes) may also be useful for describing services. Representation and indexing of moving-point objects is discussed in detail in [76, 88]. Service routes (e.g. a bus route), and service regions (e.g. airports) will require locative representation.

Filtering is sometimes applied to limit the locative availability of a service to some requestors. Some examples of filtering include calling a phone number that redirects the requestor to the appropriate provider in your region, or franchises that operate only within a specific suburb(s).

2.5.2 Price

The styles presented here describe the charging technique applied by a service provider for the use of its service. A number of styles are identified: (1) per service request or delivery (e.g. a fixed price local telephone call); (2) by unit of measure and granularity (e.g. by length, volume, weight, area or time); (3) on a percentage or ratio basis of some aspect of the service (e.g. by commission); and (4) as a range

over two values of either style (1) or style (2).

Service providers may use an aggregation of pricing styles. An example of this is a telecommunications provider (e.g. AT&T, Deutsche Telecom). The services of a telco are priced using multiple styles. This includes granular services such as per minute or second phone calls (either interstate, international or mobile phone) and per month line rental. Charges such as the initial connection fee and fixed cost local phone calls are charged on a per service request basis.

Prices typically are presented with a validity period. This may be a fixed temporal interval, or even a temporal duration beginning at a particular temporal instant. Validity within our work allows a series of prices to be identified for a service. For example, an accommodation service has distinct prices for different times during the year (e.g. school holidays, Christmas break is more expensive).

Some providers may be willing to state their negotiability with respect to price. Alternatively, it may be necessary for some providers to state that a customised price is required for a particular service (e.g. a landscaping job).

Associated with a price for a service is sometimes the need for a relationship with the service provider. We refer to this as a relationship obligation. Relationship obligations typically attempt to lock-in a service requestor to a particular provider for a specified period. Additionally, the service provider may wish to state discounts that are available with a price for a service (e.g. age based discounts).

Sometimes the charge for a service is redirected to another entity. An example of this is a free web-based email service. No cost is applied to the service requestor but advertising is used to pay for the service.

2.5.3 Payment

Payment is a process that reflects an obligation of the requestor, in response to service provision by the service provider. The payment process is normally defined by the provider, and is included as part of their business model.

Payment obligations may be required at any stage (e.g. upfront, in arrears, staged instalments) in the service provision process. These obligations are normally outlined to the service requestor as part of the negotiation process and are included in any attached settlement contracts. Service providers or their surrounding environment determine a valid set of payment instruments that are used to fulfil this important obligation of the service requestor. Payment instruments are used within the context of a payment model. The entities and information flows associated with payment models have previously been outlined in [75]. Additionally we recognise that payment protocols (e.g. Society for Worldwide Interbank Financial Telecommunications [94]) are sometimes used as a mechanism for controlling the flows within these models.

We consider the term payment instruments to be relatively self-explanatory. Such instruments include cash, cheques, direct funds transfers, credit or charge cards, travellers' cheques, wire transfers, postal or money orders, securities (i.e. stocks, options, warrants), bank bills, vouchers, stored value cards, digital cash and anonymous cash. A useful summary of payment instrument dimensions is provided

in [60].

Service providers sometimes trigger the payment obligation from the service requestor by using a request for payment or an invoice. This may indicate that the service provider has completed its obligations.

Settlement models

Two well-known settlement models are the *transactional* and the *rental* models. The transactional model can be described simply as delivery versus payment. It can be a one-off delivery or include multiple deliveries of the same service. The latter implies a longer term relationship. The rental model is the familiar concept of being "on loan" (e.g. a video). Within the rental model, explicit temporal or locative constraints may be imposed by the service provider (e.g. (a) the video is to be returned by 6pm tomorrow, or (b) when hiring a conference centre the service is found at a physical address). Depending on the service, rental may involve a short-term relationship (e.g. holiday unit) or long-term relationship (e.g. local video store membership).

Specialised forms of the transactional model are (1) *subscription*, which normally implies a long-term relationship; (2) *metered*, which is almost identical to the basic transactional model, tracks consumption of the service except that the relationship may also impose restrictions making it difficult to change to another service provider; (3) *facilitated*, in which the provider acting as a conduit or facilitator to another service provider (e.g. broker or financial planner); (4) *escrow*, which is used when there is an identified trust issue, and where the parties lodge their obligation with the escrow organisation; and (5) *swap*, where the parties agree that the services being traded are of equal value, and no payment is involved.

2.5.4 Discounts and penalties

Discounting is a common approach for attracting custom. Various types of discounts are available for services. We view discounts from the perspective of the service requestor, and therefore we believe that discounts can be categorised according to how you pay (e.g. early payment, coupon used), as well as to who you are (e.g. an elderly person). We refer to this distinction as payment related discounts and payee related discounts respectively.

The service provider is unable to determine in advance all the combinations of service discounts that might apply to a price based on attributes of the service requestor (e.g. their age, membership to associations). For this reason, the catalogue provider (who may have more context related information with respect to the service requestor) may apply the discounts to a price before it is presented to the requestor during the discovery process. We therefore consider our notion of discounts to not be included within the price specified.

We consider that a service provider might want to state the discount in one or two ways: as a reduction of the price of a service, or as a resulting price for a service. In either case, it may also be a different service where the discount is available. This

allows a service provider to entice a service requestor with a discount on the price of a service, and to then provide a discount on another (possibly more expensive) service of the provider.

Penalties are a mechanism for service providers to describe what will occur in the event that a service requestor does not comply with a specific obligation. Penalties are commonly outlined in service level agreements as a means of compensating the service requestor for non-performance (in the generic use of the term performance). An example of a condition under which a penalty is applied is for non-payment or late payment by the service requestor. Penalties will normally have a related set of conditions.

We provide for the following types of penalties: termination, financial, involuntary suspension and loss of right penalties. By termination we refer to the service provider ceasing to provide to the service requestor the output of the service. Termination is non-reversible. Our work with respect to penalties introduces a link between penalties and rights.

2.5.5 Obligations

In order to capture the responsibilities of both the service provider and the service requestor we offer the notion of "obligations" as a means of ensuring that these non-functional properties are available for discovery by interested parties. Obligations can be attached to either the request for a service, or the provision of a service. For example, a service provider may wish to advertise that service requestors have an obligation for a relationship, or an obligation to make payment should they request their service. It is the service provider who must fulfil the obligations relating to the service provision.

As one or more providers may be involved in the provision of a service, we represent the obligations of the service provider separately to service requestors. We don't associate obligations with individual service requestors, as it is unreasonable to expect a service provider to identify all individual service requestors to whom the obligations apply.

Our work captures three obligations: pricing, payment, and relationship obligations. In future, other obligations may be added to further increase the expressiveness of service descriptions, or to provide domain specific extensions with respect to obligations.

2.5.6 Rights

Provision of goods usually results in a change of ownership from the service provider to the service requestor. Services don't involve a transfer of ownership. Service providers typically own the intellectual property associated with the provision process. However, service requestors *do* have a limited set of rights that are associated with a service. These rights provide a degree of control over the request and consumption of the service.

The rights available to service requestors with most services include the following. *The right to comprehend:* service requestors should be able to question the provider with the intention of better understanding a service. *The right to retract:* once an advertised service offer has been refined into a service contract, via negotiation between the service provider and the service requestor, the service requestor can choose not to request an instance of that service. The service requestor maintains the right to request the service from another service provider. *The right of premature termination:* requestors may have the ability to prematurely terminate a service. The service provider may continue provision of the service (e.g. a movie continues to play if you get up and walk out) and may choose to apply some form of penalty for partial consumption. The latter is common in the mobile phone industry where penalties apply for early termination of mobile phone plan contracts. *The right of suspension:* interrupting the delivery and therefore the consumption of a service can act as a useful method for extending the service provision process. An example of a suspension is asking the milkman to not deliver while you are on holidays. Correspondingly, *the right of resumption:* continues the delivery and consumption of a previously suspended service.

Recourse is available in some cases to either the service provider or the service requestor. In cases where obligations of either party are not realised there may be some level of re-negotiation performed. A contracting protocol that includes the ability to decommit is outlined in [89].

2.5.7 Quality

Service quality is a measure of the difference between expected and actual service provision. It is a complex and largely domain-specific property. From the viewpoint of the requestor, it measures the competence of the provider to deliver a service [86, 58]. The most notable work on measuring customer perceptions of service quality is SERVQUAL [74]. This work produced scale that measured perceived service quality along five dimensions: the dependability and accuracy of the service (*reliability*); the promptness and the willingness of staff to assist (*responsiveness*); attributes, such as knowledge and courtesy, of staff that conveyed trust and confidence to the user (*assurance*); the level of caring and personalised attention provided to the requestor (*empathy*); and concrete or physical aspects of the service, such as cleanliness (*tangibles*).

Service providers may commit to providing a certain level of quality. This commitment is sometimes formalised using a *Service Level Agreement*. Service level agreements can be considered as binding contracts that are agreed between a service provider and service requestor. Penalties are normally imposed for non-compliance. Commitment to a service can be bound into the contracting protocol [89]. This offers a method of backing out of a service, assuming that the agreed penalty is paid. Service providers also use guarantees or warranties to express commitment to a service. A useful survey of service quality frameworks is outlined in [6].

2.5.8 Security

Security is increasingly being viewed as a mandatory component for facilitating electronic commerce. It alleviates concerns relating to identity, privacy, alteration and repudiation of information transferred between parties [18]. We commonly think about "on-the-wire" security that pertains to the request and delivery locations of a service, especially when the payment obligation of the service requestor is being finalised. Security protocols such as the Secure Sockets Layer are widespread for this role. Common approaches to security within organisations involve the implementation of a Public Key Infrastructure (PKI).

We believe that individual aspects of service descriptions should be secured. Think of a service provider who provides distinct descriptions for retail and wholesale clients (e.g. the wholesaler's description would normally include a different price). This concept is similar to visibility rules in [103]. Alternatively, multiple advertisements could be generated by a service provider with access controls applied based on the type of requestor accessing the information.

Security becomes a decidedly more complex property in the context of sub-services. We propose the following questions. (1) When a client interacts with a service and authenticates it, should they also authenticate all the sub-services? Do we require security certificates that validate aggregations or compositions of sub-services? (2) How do you secure a service to stop it from being composed within another service? Securing the discovery of the service may be an alternative [29]. (3) What are the implications for a service when some sub-services require security and others don't? (4) What happens when sub-services have differing policies with respect to client information? How do you express the security surrounding the client information to the service requestor? (5) What constitutes an infringement to a security promise? How are infringements managed (e.g. penalty payment, removal from a composition)?

2.5.9 Trust

It is easy to become very philosophical when discussing trust. As humans we use trust in a subjective manner for almost everything we do. A useful discussion of trust is offered in [62], where it is suggested that trust is a reinforcing attribute that balances perceived risk, cost and benefit. These same concerns are present in the service provision process.

Trust can be both mutual (i.e. a service provider doesn't trust the service requestor and vice-versa) and exclusive (e.g. the service provider trusts the service requestor but the service requestor doesn't trust the provider). A model for information flow within systems where mutual distrust is present has been offered in [64]. Service requestors largely view trust from two perspectives: whether they trust the intentions of a service provider and whether they trust the competence of a service provider.

Reputation mechanisms are an attempt to embody trust. Two such mechanisms have been offered to address the issues of misrepresentation and alteration in

electronic marketplaces [122]. The implementation of reputation mechanisms may be useful but concepts from non-electronic service provision may also prove useful. People tend to be satisfied that when acting within a group they will be able to increasingly trust a service provider.

The following questions arise with respect to trust in service provision. (1) How do you represent the trust of service providers or service requestors within a particular context? This question arises from a definition of reputation – "the amount of trust inspired by a particular person in a specific setting or domain of interest" [122]. (2) In a decentralised system how is knowledge relating to trust distributed, particularly changes to the perception of trust for a party? (3) How do you trust a composition (e.g. service A is composed from sub-services X, Y and Z)? Can an external party validate a service and provide a level of reputation based on previous interactions? (4) What are the implications or penalties for parties that are distrustful? (5) Does access to the past performance of a service provider reduce the perceived risk of the service requestor?

2.6 Service provider

We now present our first model that relates to the service provider [see Figure 2.1]. We stipulate that service providers have functional offerings (i.e. services) for service requestors. The function offered is referred to as a "Capability" [66] and each service provides one capability to requestors. It should be noted that this one-to-one mapping of capability to services differs from [66]. This is in large part due to the difficulty of specifying various non-functional properties for distinct capabilities within a service, if multiple capabilities were offered. We consider the area of capabilities to be the boundary of our work with respect to the functional perspective of services. We are not attempting to provide a functional description of a service. Our one to one relationship between service and capability is largely motivated by a reduced specification effort and an attempt to compel service providers to describe their services in a fine-grained manner. Introduction of a one to many relationship enforces the need to offer specialisation mechanisms that cater for inheritance of non-functional properties such as service availability, price and payment. If inheritance were to be available, non-functional properties would need to be able to be specified at both the service and the capability level. This aggregation of capabilities into services must then be sufficiently expressive to cater for overriding and exclusion of non-functional properties between the service and capability.

We envisage that a service provider will probably be an organisation (in the generic sense of the word), but may alternatively be multiple organisations. Providers are internally identified by a unique identification scheme. Dun and Bradstreet's D-U-N-S number is one possibility for this internal identification scheme [34]. Providers utilise a name for general identification. These provider names are generally granted to a provider by a regulatory authority. It is included in our model since organisations are more commonly referred to using a provider/company name (e.g. Microsoft Corporation, Deutsche Bank). We consider each service to operate within one or

more service industries that are identified according to a United Nations Standard Products and Services Code (UN/SPSC) [105].

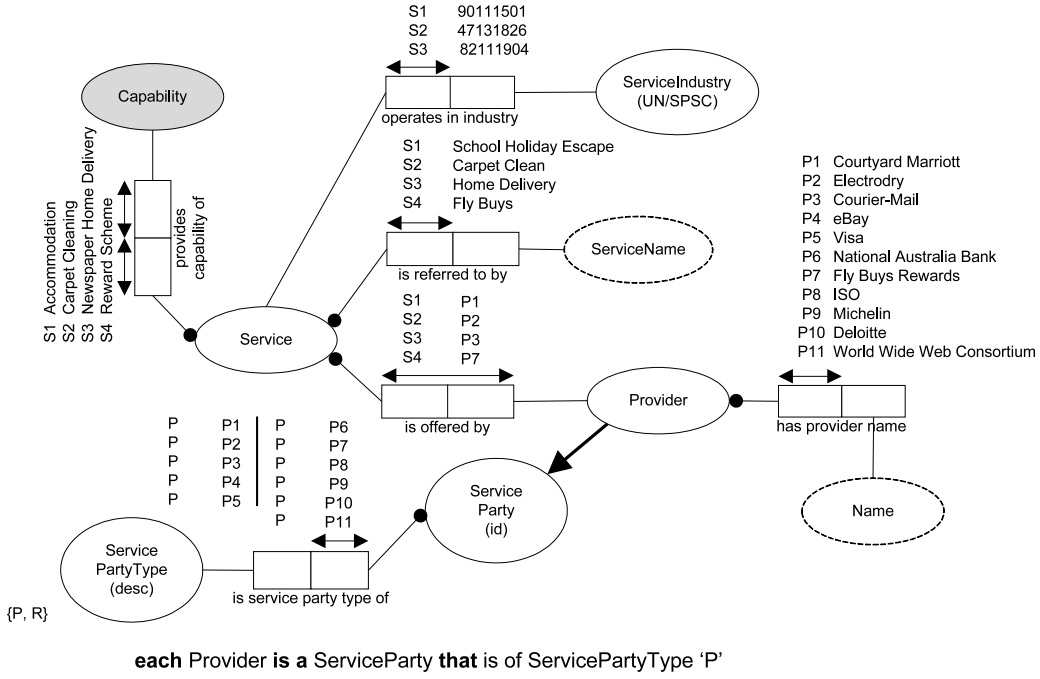


Figure 2.1: Service provider

Services are referred to by their service name. Names for services are normally intended to be unique across service providers. However, they are sometimes duplicated across services in different domains. A service name is normally determined by the service provider. The combination of a service name and a provider identifier is normally sufficient to identify a service. An exception to this is when multiple service providers are involved in the provision of a single service.

Figure 2.1 presents a single formal subtype definition (i.e. each Provider is a ServiceParty that is of ServicePartyType 'P') that is an ORM mechanism for determining membership of subtypes, using the service party type role on the ServiceParty supertype. The additional enumeration constraint of 'R' is used to define a Requestor service party. As there are no specific roles to be attached to the Requestor subtype it is not depicted as an entity in the diagram. We include it here for use in a later discussion with respect to rights.

Summary

This chapter has provided some conceptual foundations for the remainder of the thesis. In particular, we presented the notion of service, the service interactions that occur, our view of functional properties, our view of non-functional properties and how these non-functional properties will be used within the service lifecycle. In addition, we offered a set of categories of non-functional properties, including a

discussion surrounding the issues with the capture of these types of non-functional properties. Finally, we introduced the notion of service provider within our work.

The next chapter is a discussion of temporal and locative availability.

Chapter 3

Availability

This chapter is a discussion of availability with respect to services. We capture the availability of both conventional and electronic services. Our view is that availability consists of both the temporal and locative differentiators that are key to the decision making process of service requestors. First we present the base temporal and locative models. These provide the means by which we can then describe the full range of service availability. We additionally provide a short discussion of the description of communication with a location that is either for request or provision.

The entities contained within our temporal and locative models are reused throughout our conceptual models. This highlights their key nature in the description of all non-functional properties, not just service availability.

3.1 Temporal model

Our temporal model acts as a foundation for capturing when a service is available. This section attempts to define the types of temporal concepts that will be required for service description. Temporal concepts are regularly used within service descriptions to represent such things as when a service can be requested, provided, validity periods or when it can be queried for further information. Examples of temporal descriptions include:

- Newspaper delivery - A newspaper provider is offering a home delivery service for two newspapers. This home delivery service can be requested from 7am to 8:30pm Monday through Friday, and 9am to 2pm Saturday.

- Accommodation - A hotel on the Gold Coast (Australia) advertises a special or discounted rate. The rate is applicable to the accommodation being provided between 17th September and 10th October 2004 (inclusive).

- Entertainment - A seafood festival is provided from 5:30pm to 12 midnight on Friday 27th August 2004. It is part of a larger festival which occurs in Brisbane (Australia) from 27th August to the 4th September 2004.

- Home building - A building company offers a display village in five locations around a large city. The first location is open from Thursday through Sunday 10am to 5pm, and Monday 2pm to 5pm. The second location is open from Tuesday through Sunday from 10am to 5pm, and Monday 2pm to 5pm. The third location is open from Thursday through Sunday 10am to 5pm, and Monday 12pm - 5pm. The remaining two locations have the same opening times as the second location.

We collectively refer to all types of temporal concepts as "temporal entities" [see Figure 3.1]. We provide for the description of four primary types of temporal entities: dates, times (both anchored and recurring representations), intervals, and durations. We provide a discussion of these types in the sub-sections that follow. Firstly we present dates and time as they act as the basis for the description of instants, which in turn are used to describe intervals.

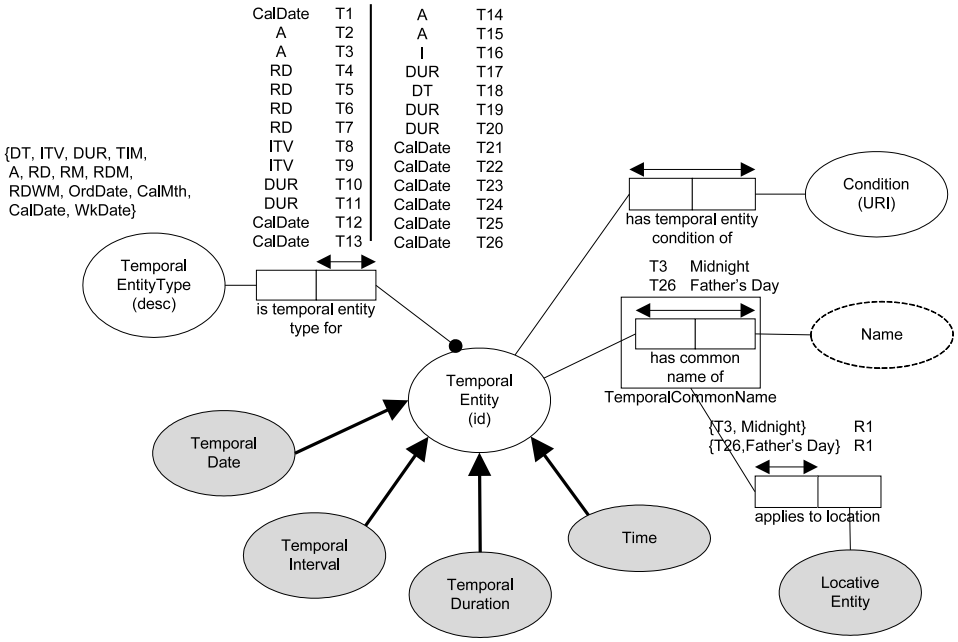


Figure 3.1: Temporal entities

We provide for the expression of a common name for a temporal entity. This allows us to express a temporal concept such as an interval (e.g. a particular week of the year) and apply a common name to assist with service discovery. For example, in Australia the first Sunday of September is Father's Day. Our temporal model provides for the expression of the temporal interval (see instance number T26), the first Sunday in September (more details are provided later). To this temporal entity that we describe we can then attach the common name "Father's Day". Additionally, we offer the ability to put this temporal common name into a locative context. This locative context is likely to be a region such as a country or state.

The ORM populations depicted in Figure 3.1, particularly those surrounding the subtype defining role ("is temporal entity type for"), are used as the basis for subsequent model populations.

3.1.1 Dates

We classify dates into four subtypes - calendar dates, ordinal dates, week dates and calendar months [see Figure 3.2]. The first three of these are outlined in [49]. Our discussion of dates assumes the use of the Gregorian calendar. We do not propose to support other calendars. The four date subtypes can be defined as:

- *Calendar dates* are described using a year number, a month number between 1 and 12 (representing the months of January through December) and a day of month number between 1 and 31. For example, a year number of 2004, a month number of 8 and a day of month number of 27 represents the 27th August 2004.

- *Ordinal dates* are a combination of a day of the year number between 1 and 366 (catering for leap years) and a year number. For example, a day of the year number of 240 combined with a year number of 2004 represents the 27th August 2004.

- *Week dates* are defined using a day of week number (where 1 - 7 identifies the days Sunday through Saturday), a week of year number (indicated by a number between 1 and 52), and a year number. For example, a day of week number of 6, a week of year number of 35, and a year number of 2004 represents the 27th August 2004.

- *Calendar month dates* are a subset of the properties of calendar dates, and are defined using a month of year number and a year number. For example, a month of year number of 8 and a year of 2004 represents August 2004.

Figure 3.2 includes a number of ORM external uniqueness constraints, each of which stipulates uniqueness across two or more fact types. The external unique constraint is depicted as dotted lines joined to a circled "U". For example, the combination of a year and day of year number create a unique Ordinal Date. This same figure also includes a derived role. The role "is a leap year" is attached to the Year entity. The "*" denotes that it is derived. In this case, the derivation is based on a well-known algorithm to determine if the year is a leap year or not.

Figure 3.2 presents five formal subtype definitions (e.g. **each** WeekDate **is a** TemporalDate **that** is of TemporalEntityType 'WkDate') that provide a means for determining membership of subtypes, using the temporal entity type role on the TemporalEntity supertype. We have chosen not to present any subtype definitions on Figure 3.1. Instead we list all subtype definitions on the figure that contains the detail for the entity.

We make use of ConQuer, a conceptual query language, within this paper as a means of providing examples of conceptual queries that could be applied to the ORM schemas. A detailed discussion of ConQuer is presented in [12]. The following is an example ConQuer query over week dates that returns all the TemporalDate instances (subtype of TemporalEntity) where the year is 2004, the week number is 39, and the day of the week number is 6. Assuming that the first week of the year

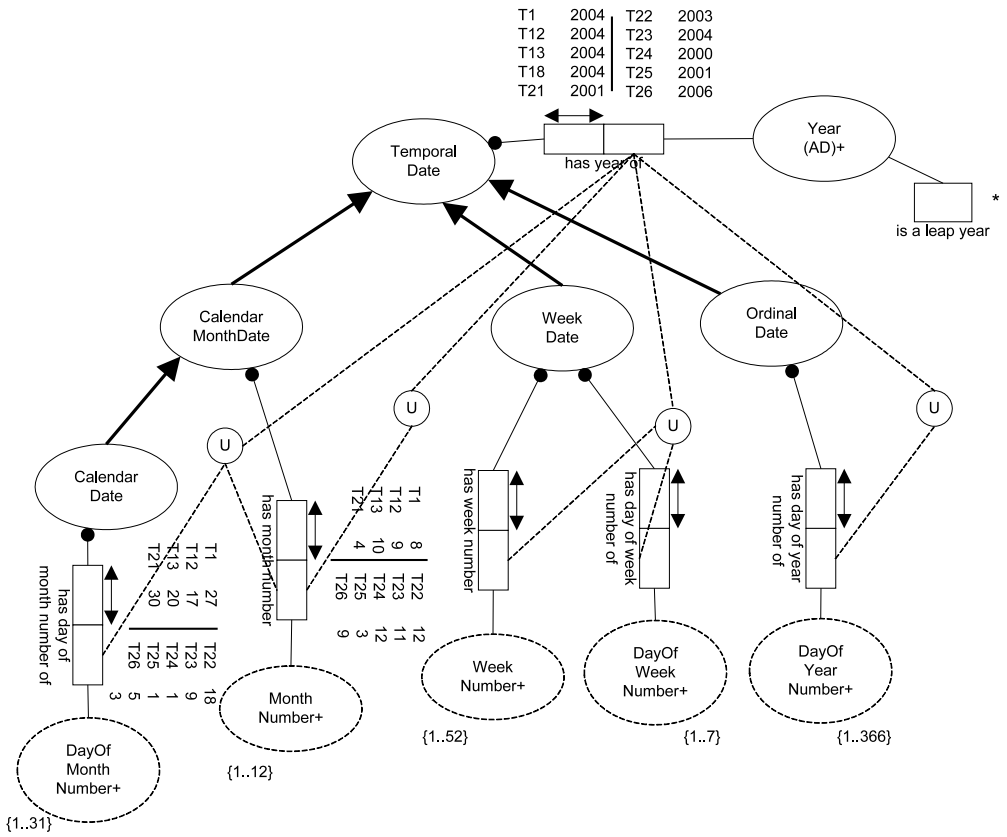


Figure 3.2: Temporal dates

2004 starts on Sunday 4th January, then the following query returns all instances equal to the 1st October 2004.

TemporalDate
- ⊢ has year 2004
- ⊢ is WeekDate
  - ⊢ has week number 39
  - ⊢ has day of week number 6

The following is an example query over calendar dates that returns all the TemporalEntity instances where the year is 2004, the month number is 8 and the day of the month number is 27. This returns all instances equal to the 27th August 2004.

TemporalDate

⊢ has year 2004
⊢ is CalendarMonthDate
⊢ has month number 8
⊢ is CalendarDate
⊢ has day of month number 27

Later in this chapter we present a discussion of temporal instants. It is important to note that each of the date subtypes except for calendar month dates results in a temporal entity that has a granularity of "day". A calendar month date results in a temporal entity with a granularity of "month". The distinction is raised here as we are unable to use the calendar month date subtype to build a temporal instant.

3.1.2 Time

We represent the concept of time so that we may describe points in time. By time we refer to the time of day that is displayed on a clock. When combined with other properties (e.g. a calendar month date, a day of the week) we consider points in time to be either anchored (occurring only once), [see Figure 3.3] or recurring (occurring more than once, normally with some regularity). Time is particularly useful for the description of intervals (either the start or the end of an interval in combination with a duration, or jointly), and as a deadline (e.g. the closing submission time of a tender).

Later, when used to describe a temporal interval, points in time act as boundary positions (i.e. the start or end position) of an interval (see section 3.1.3). Accordingly, we believe that we achieve a similar notion of anchored versus recurring temporal intervals through the use of recurring points in time. We do not attempt to describe time with a granularity smaller than seconds (e.g. milliseconds). For other parts of the model that describe temporal durations we provide for the ability to define granularities of less than a second. We therefore say that our time representation cannot be further divided into a smaller unit of time.

We consider all times to have at least the following properties: hours (a unit of 60 minutes), minutes (a unit of 60 seconds), seconds (the smallest unit of time that we choose to represent), and a timezone (expressed as a positive or negative offset from -12 to + 12). This offset is expressed according to Coordinated Universal Time (UTC), and includes hours and minutes (e.g. Australian Eastern Standard Time is UTC +10:00). A full list of UTC timezones is available [101]. In addition to the offset we capture a region that the timezone relates to, an textual abbreviation and a full name. The region is captured via a type of locative entity called "Region" that will be discussed in more depth in section 3.2.3.

As mentioned previously, we offer the ability to store a common name for all temporal entities. The usefulness of this may be seen in the ability to describe a UTC time of 00:00:00+10:00 using the common name "midnight", whilst 12:00:00+10:00 could be given the common name "midday".

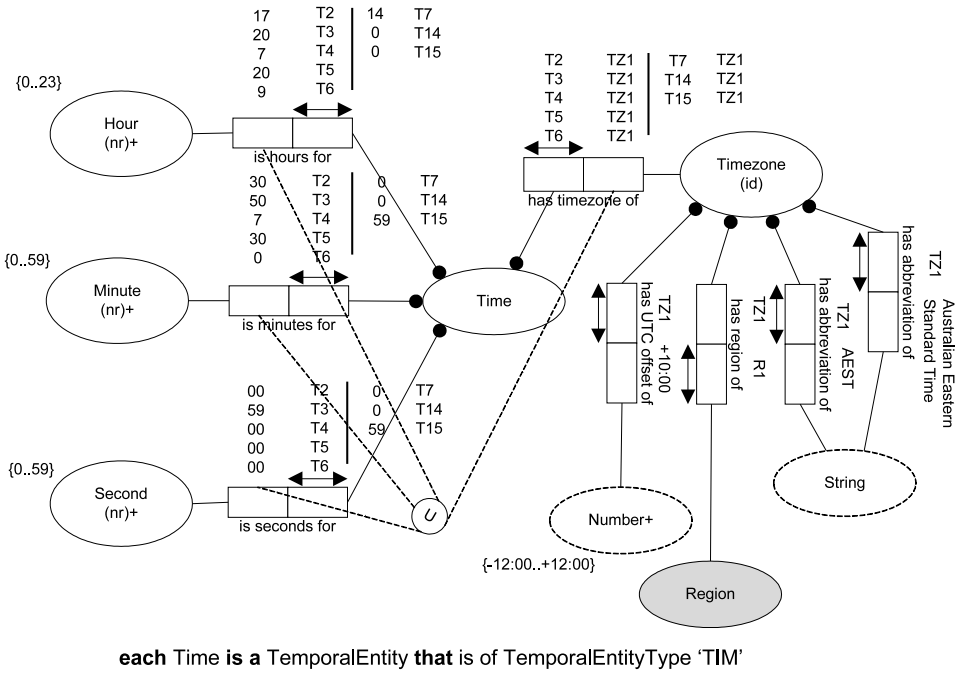


Figure 3.3: Time

Points in time: We believe that time has five specific subtypes, the most important of which is the anchored point in time. This will be discussed shortly. Other subtypes of time include recurring daily time, recurring day of month time, and recurring day of week in month time. Examples of these recurring time subtypes include:

- Recurring Daily Time in a Week - Day of week number and a time. For example, a day of week number of 2 and a time of 14:25:00+00:00 represents Monday at 2:25pm.
- Recurring Daily Time in a Month - Month number and a time. For example, a month number of 8 and a time of 06:00:00+00:00 represents every day in the month of August with a time of 6:00am.
- Recurring Day of Month Time - Month number, day of month number and a time. For example, a month number of 11, a day of month number of 27 and a time of 18:30:00+00:00 represents November 27th at 6:30pm.
- Recurring Day of Week in Month Time - Occurrence number, day of week number, month number, and a time. For example an occurrence number of 1, a day of week number of 2, a month number of 11 and a time of 09:45:00+10:00 represents the 1st Monday in November at 9:45am (with a UTC offset of 10 hours).

Anchored points in time: Anchored points in time or anchored temporal instances are fixed by inclusion of a date with the time [see Figure 3.4]. As stated

previously, anchored points in time occur only once. This subtype allows us to capture requirements such as due dates (e.g. for payment) and to build temporal intervals using a start and end anchored point in time. All Time subtypes inherit all the properties of the Time temporal entity (i.e. hour, minute, second and time-zone). The attachment of a date to a time produces a temporal instant. Importantly, temporal instants include a date where the type of date used has a "day" level of granularity, not a particular month as is the case with calendar month date types. See section 3.1.1 for a discussion of temporal dates.

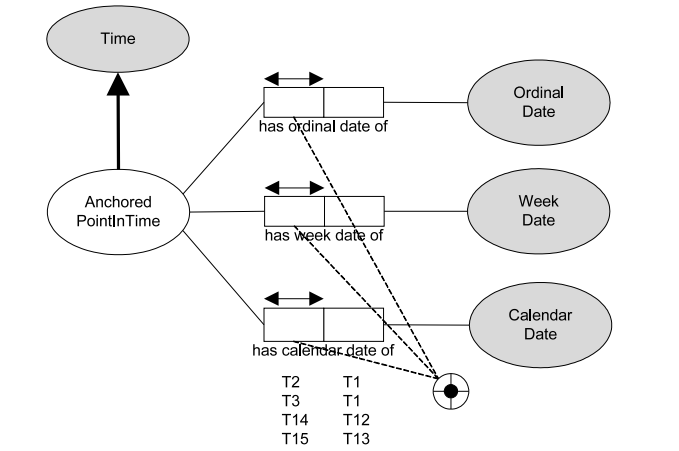


Figure 3.4: Anchored temporal instants

Figure 3.4 includes an ORM exclusive-or constraint that stipulates only one role is played. The exclusive-or constraint is depicted as dotted lines joined to a circled "X" with a mandatory symbol (solid black dot) centred over the "X".

The following is an example ConQuer query over an anchored point in time. The query returns all the times where the time is 8:30am on the 6th February 2006 in any timezone that is specified using a CalendarDate.

```
AnchoredPointInTime
    ⊢ has hours 8
    ⊢ has minutes 30
    ⊢ has seconds >= 0
    ⊢ has seconds <= 59
    ⊢ is CalendarDate
        ⊢ has year 2006
        ⊢ has month number 2
        ⊢ has day of month number 6
```

Recurring times: We refer to all recurring subtypes as not being anchored in time, and can therefore say that they apply at multiple anchored points in time over

an anchored temporal interval [see Figure 3.5]. All subtypes inherit the properties of the time entity (i.e. hour, minute, seconds and timezone).

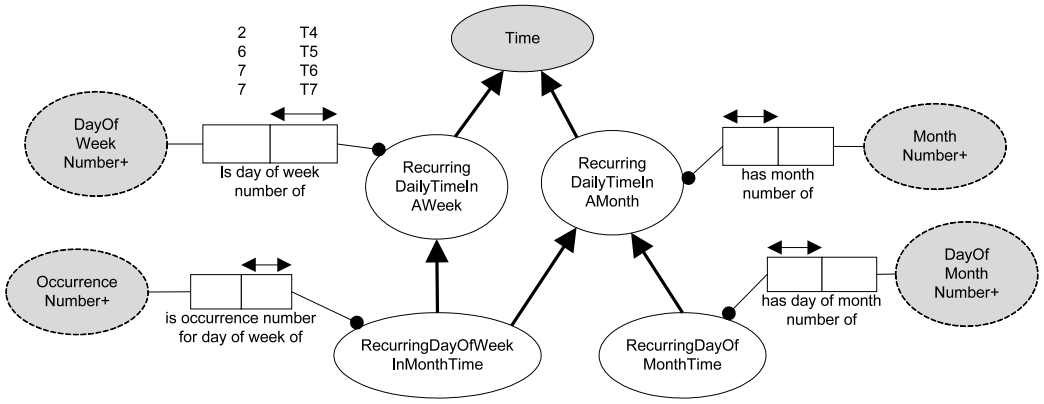


Figure 3.5: Recurring time

As its name suggests, recurring daily time in a week is useful for describing a time that occurs on a daily basis. For example, Monday 9am might be used to describe the time that provision of a service regularly starts. In the newspaper home delivery example provided above, the service was able to be requested from 7am to 8:30pm Monday through Friday, and 9am to 2pm Saturday. We may also choose to think of this as being 7am - 8:30pm Monday, 7am - 8:30pm Tuesday and so forth until Friday, and then 9am to 2pm Saturday. Each case is an interval that can be demarcated by a start and an end time that is of a recurring daily time in a week type.

Recurring daily times in a month specify a month number. This means that the recurring nature of this time subtype is daily within the month specified. For example, a chateau holds a festival daily in February from 11am to 5pm.

Recurring day of month times specify a day of the month, and a month. For example, a local church may offer a Christmas mass where the service is provided each year on the 25th December at 6:00am. This means that the recurring nature of this subtype is annually.

Recurring day of week in month times specify the occurrence of a day of week, within a particular month. For example, a single parents support group may meet on the 1st Monday of each month at 7:00pm. The corresponding population of this subtype requires that each of the twelve months of the year be specified (i.e. 1st Monday of January at 7:00pm, 1st Monday of February at 7:00pm through to the 1st Monday of December at 7:00pm).

3.1.3 Temporal intervals

We refer to intervals as being either anchored or recurring. Providers historically have used recurring intervals to describe the availability of their services. The length of time between the start time and the end time of an interval we refer to as the temporal duration. In the case of a recurring interval we consider the duration to be the length of time between the start and end of a single temporal interval instance within the recurring interval. We express temporal intervals using one of three different approaches [see Figure 3.6]. Each of the following approaches are outlined with an example of anchored points in time:

- From a Time to another Time: This type of interval is defined with a specific start time and end time. For example, a start time of 9:00:00+10 on calendar date 12/10/2005 and an end time of 23:00:00+10:00 on calendar date 12/10/2005 defines the period between 9am Australia Eastern Standard Time and 11pm (in the same timezone). The second time that is described occurs temporally after the first time. An example of this type of interval is the entertainment service outlined in section 3.1 that began at 5:30pm on the 27th August 2004 and finished at midnight on the same day.

- From a Time for a Duration: This type of interval is described by declaring a start time and specifying a duration. For example, a start time of 8:30:00+10:00 on calendar date 20/06/2004 with a duration of 3 days. The end time can be derived by adding the duration to the start time. In this case the end time would be 8:29:59+10 on the 23/06/2004.

- For a Duration to a Time: The reverse of the previous approach can be used to describe a duration with a specific end time. Using this approach the start time can be derived from taking the duration away from the end time.

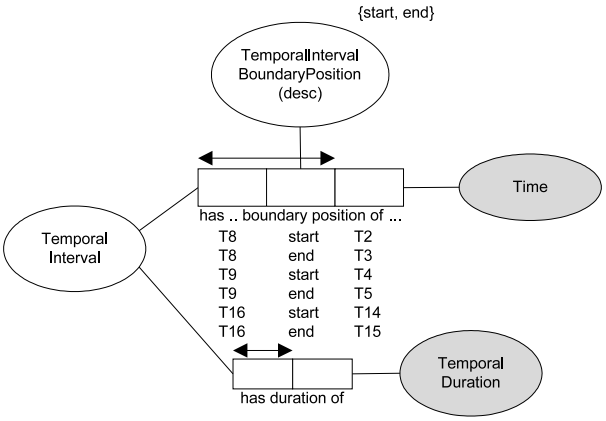


Figure 3.6: Temporal intervals

Whilst dates could be argued to be a form of temporal interval, subtyping it from the TemporalInterval entity introduces confusion when expressing anchored points in

time. A particular date (e.g. 10/03/2005 or 10th March 2005) can viewed as: time to another time (0:00:00+10:00 10/03/2005 to 23:59:59+10:00 10/03/2005), time with a duration (0:00:00+10:00 10/03/2005 with duration of 1 day), or duration to a time (1 day to 23:59:59+10:00 10/03/2005). We provide a distinct treatment of dates within our temporal model. See section 3.1.1 for further details.

The following is an example ConQuer query over a temporal interval using anchored points in time that have been defined with a CalendarDate. The query returns all the instances where the start time is 8:30:00+10:00 on the 6th February 2006, and the end time is 8:30:59+10:00 on the 9th February 2006.

```
TemporalInterval
    ⊢ has 'start' boundary position of AnchoredPointInTime
        ⊢ has hours 8
        ⊢ has minutes 30
        ⊢ has seconds 0
        ⊢ has UTC offset +10:00
        ⊢ is CalendarDate
            ⊢ has year 2006
            ⊢ has month number 2
            ⊢ has day of month number 6
    ⊢ has 'end' boundary position of AnchoredPointInTime
        ⊢ has hours 8
        ⊢ has minutes 30
        ⊢ has seconds 59
        ⊢ has UTC offset +10:00
        ⊢ is CalendarDate
            ⊢ has year 2006
            ⊢ has month number 2
            ⊢ has day of month number 9
```

Some providers prefer to use temporal interval descriptions such as 9am till late. This could be facilitated within our model by allowing the provider to specify an end instant for the temporal interval, and assigning it a temporal common name (discussed in section 3.1) of "late".

Recurring temporal intervals: Expressing recurring temporal intervals is useful for service providers who wish to regularly advertise the availability of a service without the need to update a service description. For example, the newspaper delivery service was available from 7am to 8:30pm Monday through Friday, and 9am to 2pm Saturday. We could use the notions of recurring daily time in a week presented previously in conjunction with an interval in the following way to represent this example. Remembering that this example is really the conjunction of 6 intervals (one per day of the week), the first interval start time could be represented using a

recurring daily time of Monday 7am, with a terminating time of 8:30pm Monday. Alternatively, the same start time could be represented with a duration of 13.5 hours.

We envisage the need to express the following types of temporal intervals:

- A month (e.g. December): This could be represented as a recurring day of month time of (1st December at 0:00:00+00:00 with a duration of 31 days).
- An occurrence of a day of week within a month (e.g. the 3rd Sunday in July): This could be represented using the RecurringDayofWeekInMonthTime that has the values - the month number is 7 (for July), the day of week number is 1 (for Sunday), the occurrence number for the day of week is 3 and the time is 0:00:00+00:00. With a duration of 24 hours this interval is capable of being represented.
- A day of a month (e.g. 25th December): This could be represented with a day of month number of 25, and a month number of 12, and a time of 00:00:00+00:00. Like these other examples the interval can be expressed with this start time and the duration (in this case 24 hours), the start time and an end time (23:59:59+00:00 on the 25th December) or a duration and the end time.

The following is an example ConQuer query over a recurring temporal interval such as day of a month. This query returns all TemporalIntervals that represent the recurring temporal interval of the 25th December using a start time and a duration. Alternatively, this could be queried using the common name attached to the recurring temporal interval.

```
TemporalInterval
    ⊢ has 'start' boundary position of RecurringDayOfMonthTime
        ⊢ has hours 0
        ⊢ has minutes 00
        ⊢ has seconds >= 0
        ⊢ has month number 12
        ⊢ has day of month number 25
    ⊢ has duration of TemporalDuration
        ⊢ has cardinality 24
        ⊢ has temporal granularity of StandardTemporalGranularity
            ⊢ has standard granularity name 'Hour'
```

Temporal interval operations: We provide the ability to recursively describe exceptions to intervals (i.e. exclusion of a sub-interval) and restrictions over an interval (i.e. refinement to a particular sub-interval) [see Figure 3.7]. This allows us to specify a temporal interval (e.g. June - November 2004) but to restrict to just

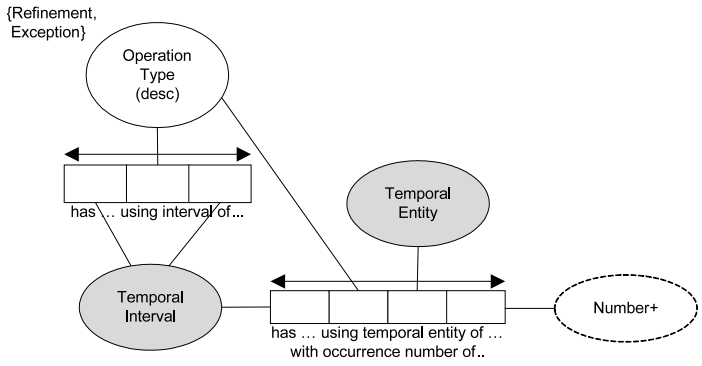


Figure 3.7: Temporal interval operations

a subset of that (e.g. Tuesday's) within that interval. Alternatively we can specify the same temporal interval and exclude intervals within that (e.g. Thursday's).

We offer an alternative to specifying the refinements and exclusions within a temporal interval. This is depicted in Figure 3.7 as the quaternary relationship whereby a TemporalInterval entity can have an OperationType (i.e. refinement or exclusion) specified as another TemporalEntity with a particular occurrence number within the interval. This allows us to perform refinement and exclusion such as, specifying the occurrence of a particular day of the week within a month (e.g. 2nd Sunday in August 2005).

We build upon our previous example query to show the use of temporal interval operations in the context of searching for exclusions to service availability (e.g. a service is not available on the 25th December). This query returns all TemporalIntervals that represent the recurring day of month time temporal interval of the 25th December as an exception.

TemporalInterval
- ⊢ has OperationType of 'Exception' using interval of TemporalInterval
  - ⊢ has 'start' boundary position of RecurringDayOfMonthTime
    - ⊢ has hours 0
    - ⊢ has minutes 00
    - ⊢ has seconds >= 0
    - ⊢ has month number 12
    - ⊢ has day of month number 25
  - ⊢ has duration of TemporalDuration
    - ⊢ has cardinality 24
    - ⊢ has temporal granularity of StandardTemporalGranularity
      - ⊢ has standard granularity name 'Hour'

Alternatively, an interval could be represented as finishing prior to the 25th December or beginning after the 25th December. The union of another query would be required to produce all intervals that exclude the 25th December.

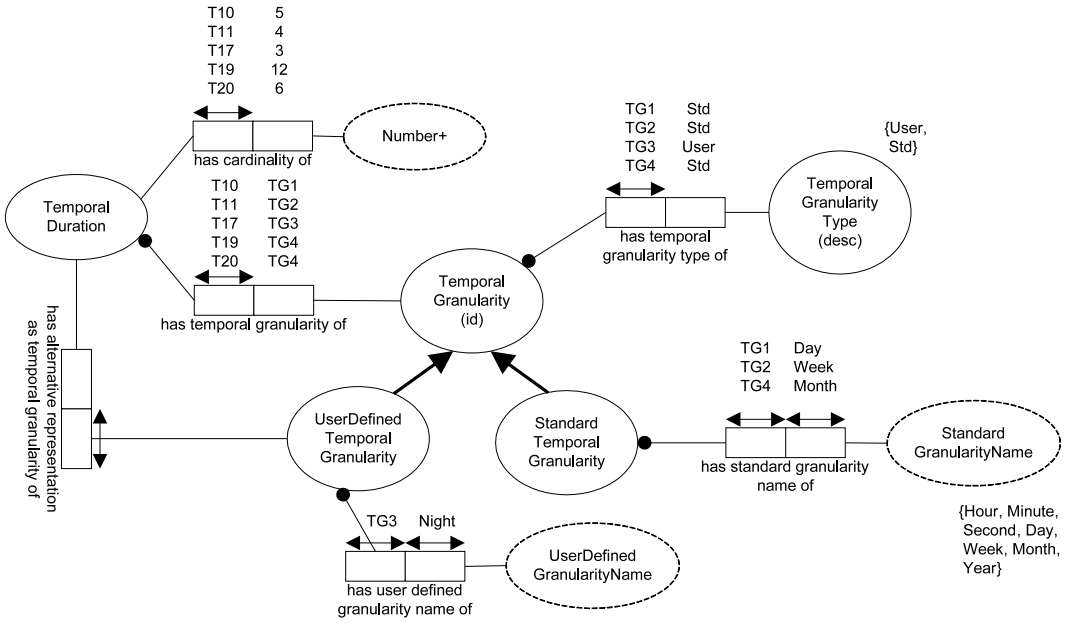
The following example query outlines the use of temporal interval operations for refinement. The query presents the refinement of an initial temporal interval (e.g. August 2005), to be the 1st Sunday in August 2005 at 10am.

TemporalInterval
- ⊢ has 'start' boundary position of AnchoredPointInTime
  - ⊢ has hours 0
  - ⊢ has minutes 00
  - ⊢ has seconds >= 0
  - ⊢ has month number 8
  - ⊢ has day of month number 1
  - ⊢ has day of month number 2005
  - ⊢ has duration of TemporalDuration
    - ⊢ has cardinality 31
    - ⊢ has temporal granularity of StandardTemporalGranularity
      - ⊢ has standard granularity name 'Day'

⊢ has OperationType of 'Refinement' using TemporalEntity with occurrence number of 1
- ⊢ is RecurringDailyTime
  - ⊢ has hours 10
  - ⊢ has minutes 00
  - ⊢ has seconds >= 0
  - ⊢ has day of week number 1

3.1.4 Temporal duration

The final type of temporal entity, temporal duration is used to express lengths of time (e.g. 5 days, or 4 weeks) [see Figure 3.8]. We describe temporal durations using a cardinality (a number), and a temporal granularity. We divide temporal granularities into one of two different types. Standard temporal granularities are temporal concepts that are readily familiar in most domains. These concepts include hour, minute, second, day, week, month and year. Alternatively, user defined granularities can be captured. These may include notions such as business days. We allow these user defined temporal granularities to be expressed in terms of another temporal duration. Temporal durations may also have a temporal common name attached through the supertype TemporalEntity. This can be useful as it allows us to specify a duration of 1 day and assign it a temporal common name of "Monday". We apply this use of names for TemporalDuration entities when using temporal interval operations (e.g. the refinement or exclusion of the "x"th occurrence of Monday within a temporal interval).



each TemporalDuration **is a** TemporalEntity **that** is of TemporalEntityType 'DUR'
each StandardTemporalGranularity **is a** TemporalGranularity **that** is of TemporalGranularityType 'User'
each UserDefinedTemporalGranularity **is a** TemporalGranularity **that** is of TemporalGranularityType 'Std'

Figure 3.8: Temporal duration

3.2 Locative model

The next group of models that we have chosen to present relates to the non-functional property of location. Our notion of the locative aspect of services is wider than just a geographic interpretation. The locative model that we present (i.e. "the where") acts as a foundation (along with our temporal models) for the availability of a service.

This section attempts to define the types of locative concepts that will be required for service description. Locative concepts are regularly used within service descriptions to represent properties such as where a service can be requested from, where it can be provided to, and where payment can be directed. Examples of such descriptions include:

- Dog minding: A dog minding service is provided in Brisbane, the Gold Coast and Cairns.

- Accommodation: A hotel in Sydney accepts requests on a published phone number, on a toll free phone number and via email using a specific email address.

- Seminar: A property investment seminar can be requested using a published phone number and is provided at a specific hotel in Melbourne (address provided).

- Home building: A builder provides five locations for display villages. The address of these villages are provided using the street address and a street directory reference. More information is available using a published phone number, or via their web site.

We collectively refer to all types of locative concepts as "locative entities". We divide locative concepts into ten subtypes: points (stationary and moving), regions, routes, addresses, phone numbers, street directory references, URIs, spectra, Internet Protocol (IP) addresses and Ethernet addresses [see Figure 3.9]. To each instance of a locative entity we allow one or more regionalised, common names to be attached. This is similar to our treatment of TemporalEntity common names.

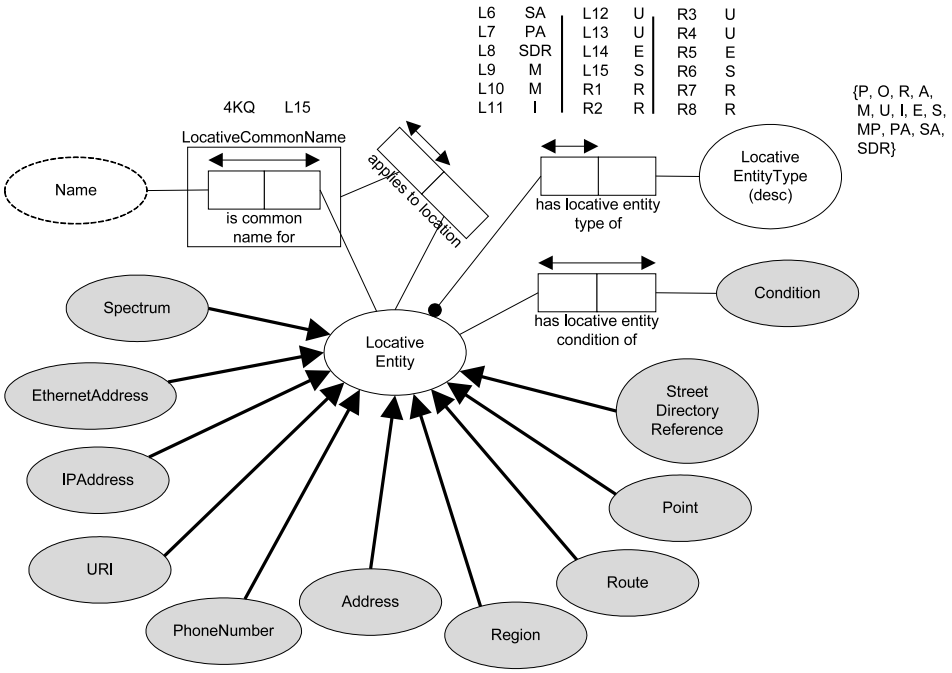


Figure 3.9: Locative entities

A location represents an important non-functional property that is not only capable of describing such things as where a service provider can provide the service, it also indicates the presence of distance between the requestor and provider. For example, electronic locative entities such as URIs, spectra, IP addresses and Ethernet addresses involve communications at "arms length". We have previously referred to this as a channel [68]. We now refer to it simply as an interaction.

The ORM populations depicted in Figure 3.9, particularly those surrounding the subtype defining role ("has locative entity type of"), are used as the basis for subsequent model populations.

3.2.1 Points

A point is a position on the Earth's surface and must be supplied with coordinates of latitude and longitude [see Figure 3.10]. An altitude may also be recorded for

that point. Coordinates are described using degrees, minutes and seconds. Degrees are 1/360th of a circle and are further subdivided into 60 minutes, and then into 60 seconds [37]. In a geographical context, these points are sometimes referred to as "waypoints" [38]. When we refer to coordinates we are not referring to celestial coordinates.

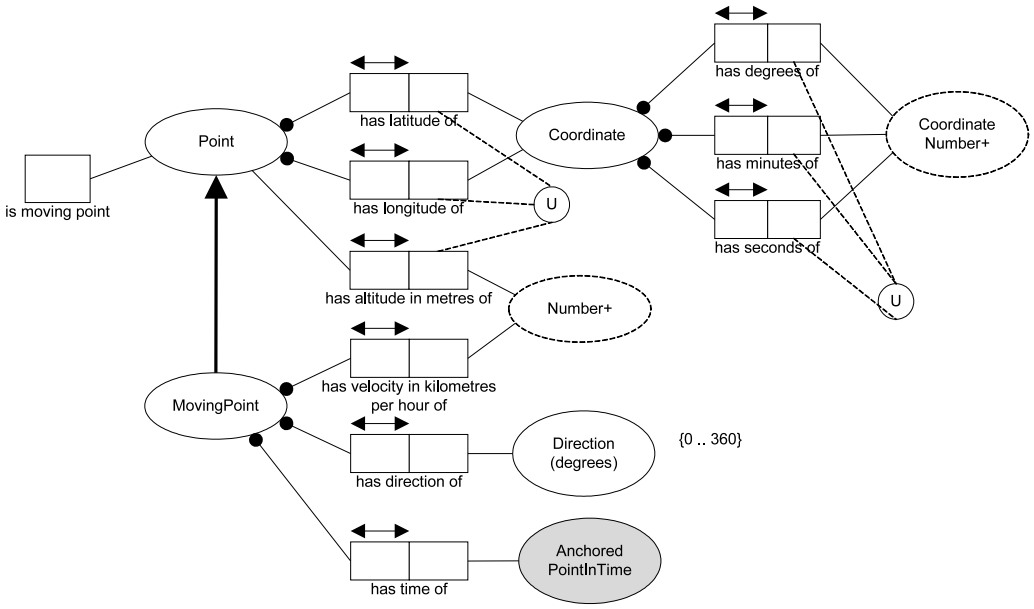


Figure 3.10: Points (including moving points)

We further subtype points to include a "MovingPoint" entity. In addition to the properties of stationary points, moving points include a velocity, an anchored point in time and a direction. Moving points are useful in the context of routes which are discussed in the section that follows.

3.2.2 Routes

We consider a route to be an ordered collection of points. To ensure that the service provider is not burdened with outlining the specific details of some route, we present an abstraction that allows the specification of a route to be optionally attached. Figures 3.11 and 3.12 present our modelling of routes. Routes are commonly used to store the locative properties of moving objects such as buses, trains and planes. To each route we assign a route type and a route name. Together, these act as a high level description of the route. The enumeration constraints presented in Figure 3.11 for route types are intended to be indicative.

We envisage that the catalogue provider may provide a base specification for major routes, and the service provider may refer to this specification. Refer to our discussion in section 1.5.1 about the onus of descriptive effort. Alternatively, they

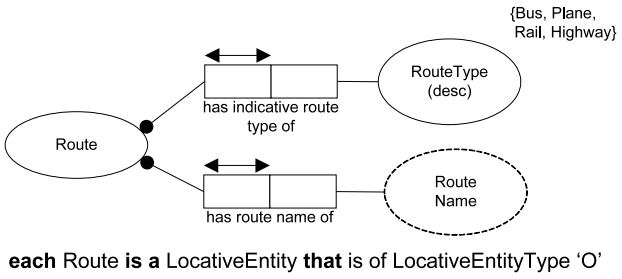


Figure 3.11: Routes

may choose to provide their own specification that is either more coarse-grained or fine-grained.

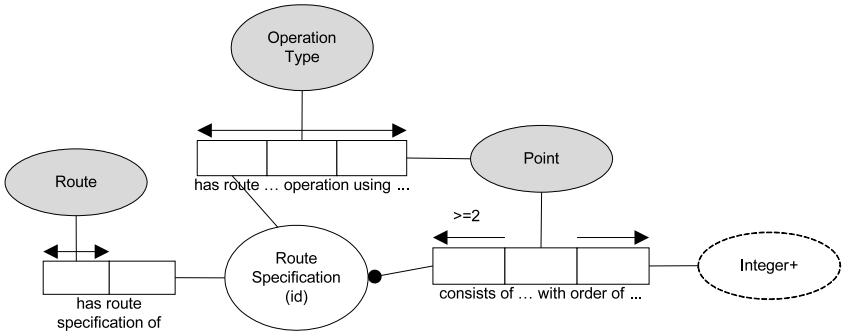


Figure 3.12: Route specification

The ordering of the points within the route is useful for determining the start and end of the route. Routes, like the other subtypes of locative entities, include one or more common names. In a manner similar to our temporal model we allow for the restriction of a route (i.e. refinement to a sub-section of the route) and for exceptions to a route (i.e. exclusions of part of a route). For example, a bus route between suburban Melbourne and the city centre may normally involve 20 stops. A "rocket" service may only utilise 10 of those same stops. This could be expressed using exclusions to the normal bus route. Since a route is capable of capturing the anchored points in time of a series of moving points, the notion is similar to that of a "schedule". Whilst we realise that some routes may involve stopping at the same point multiple times within a single invocation of the service (e.g. a route shaped as a figure of eight), we utilise the same point but provide it with a different ordering value. We apply a frequency constraint to the creation of a route that enforces the need for 2 or more Point instances to ensure the existence of a beginning and an end to the route, otherwise it can only be considered a point. It should be noted that routes can be an ordered collection of MovingPoint entities. This allows for the inclusion of directional and velocity related information in the route description.

The following example query filters instances of route specifications where the first moving point of the route is referred to by the common name "Brisbane" and where the last point of the route is referred to by the common name "Gold Coast". This query returns all kinds of routes between Brisbane and the Gold Coast (e.g.

bus, plane, train). Thereby, allowing us to find out about different ways of getting from one place to another.

RouteSpecification
⊢ is Point
  ⊢ has common name “Brisbane”
⊢ has min(ordering) for RouteSpecification
⊢ is Point
  ⊢ has common name “Gold Coast”
⊢ max(ordering) for RouteSpecification

3.2.3 Regions

Our treatment of regions is similar to that of routes. We abstract the specification of the region to reduce the burden of specification on the service provider. Figures 3.13 and 3.14 present our modelling of regions. We assign a region type and a region name to each region. Together, these act as a high level description of the region. The enumeration constraints presented in Figure 3.13 for region types are intended to be indicative.

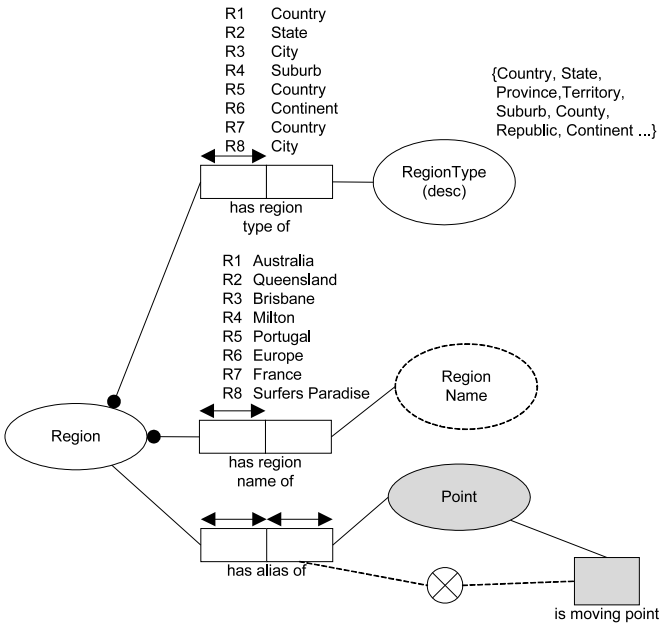


Figure 3.13: Regions

We consider a region to be a bounded collection of non-moving points which are used to describe an area. We enforce the usage of non-moving points using the exclusion constraint depicted in Figure 3.13. Regions are an abstraction that we use for capturing concepts such as countries, republics, states, territories, provinces,

counties, cities, and suburbs. Other less common types of regions include franchise areas and amusement parks. We consider RegionSpecification entities to provide the detail about the region. We allow for exceptions to a larger region to be stated, whilst restrictions (or sub-areas) may also be captured. We consider that region specifications are likely to be a part of the service catalogue that will be populated by the catalogue provider. This population of parts of the catalogue ensures that (in this instance) common regions such as countries and states are available for use by service providers.

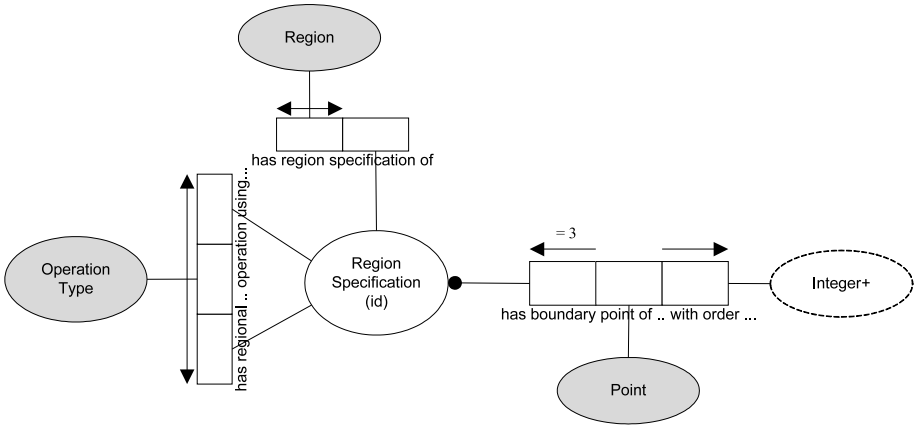


Figure 3.14: Region specifications

We assign a frequency constraint to the ternary fact type in Figure 3.14 to enforce that a minimum of three points constitute a bounded region. It is, of course, possible to produce a line (with three points along the line) using such a constraint but it is our intent that a bounded region be formed. We are unable to graphically represent in ORM that a region should be a bounded area. To form a line with multiple points we utilise the notion of a route as previously outlined in section 3.2.2.

The following example query filters for instances of regions where the region type is “Territory” and the region name is “Australia”.

Region
⊢ has region type of “Territory”
⊢ has region name of “Australia”

3.2.4 Addresses

In general, all addresses include a country, a state or province, a city, suburb, and a postcode or zipcode [see Figure 3.15]. Our discussion and modelling of addresses is based on the United States Postal Service - Postal Addressing Standards (Publication 28) [110].

To any address we offer the ability to capture information related to the party at the address to which a request is being directed [see Figure 3.16]. This addressee

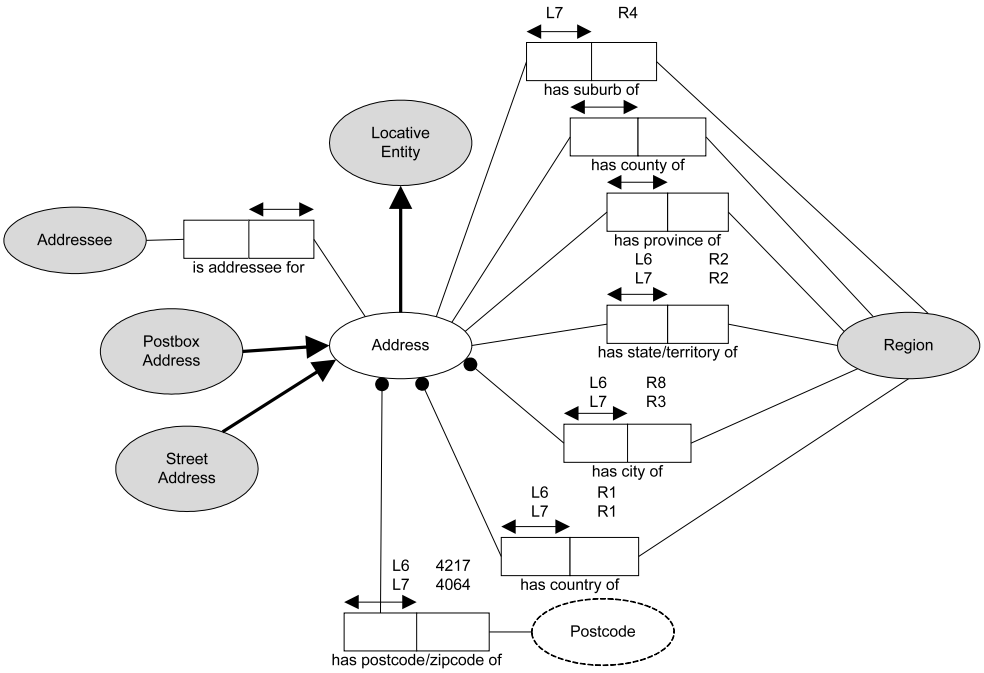


Figure 3.15: Addresses

related information such as addressee name, professional title, functional title, department name and/or organisation name are presented here, and not in the service provider model (see section 2.6) for one particular reason. When we describe services, the interactions that occur between a service provider and a service requestor can only happen at a location, or via a communication mechanism to a location. This addressee related information is specific to the address subtype of LocativeEntity. This is accurate whether the address is used to reflect a postal address or to indicate the physical presence of the service provider.

We consider that addresses are normally of two types: a street address or a postbox address [see Figures 3.17 and 3.19]. Street addresses may include the representation of a unit number, room number, apartment number, suite number, level or floor number, building number, street number, a street name and a street type. A full list of street types and their abbreviations is available [110]. The enumeration constraints presented in the model for street types are not intended to be a complete treatment, just indicative of the types of values that could be contained within this entity. We do not try to capture addresses such as the "corner of street x and street y". We take a stance similar to that presented in [110] that a number on either street x or street y will uniquely identify the location. The use of corner appears to be a term of convenience for the service requestor. It is feasible for the locative common name to include the reference to the corner. Street numbers within our model are a many to many relationship this allows us to capture a range of street numbers (e.g. 9 - 11). Street addresses may include zero or more street directory

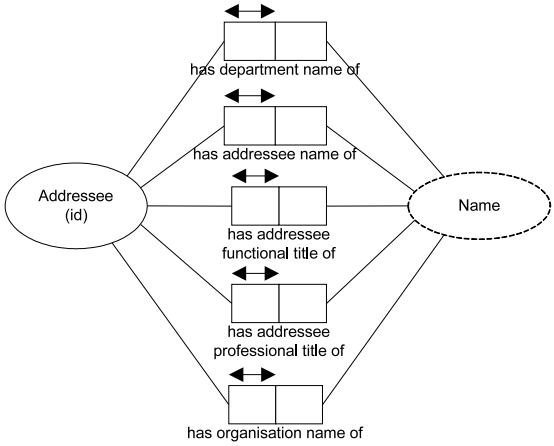


Figure 3.16: Addressees

references, considering that the address may appear in street directories published by different providers. We also allow one or more map references to be a attached to a street address. This allows services such as Google Maps to be used [44].

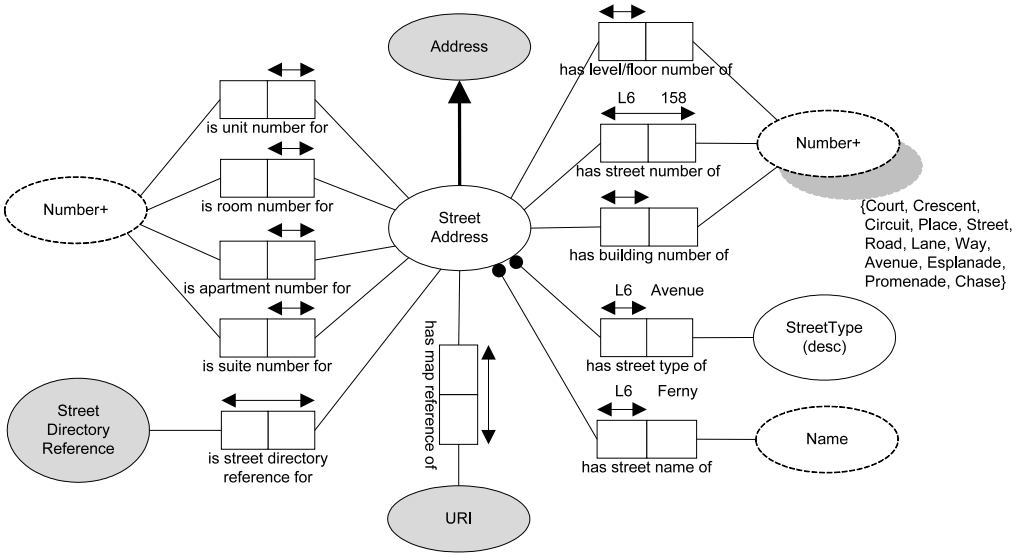


Figure 3.17: Street addresses

We refer to street addresses as being in proximity to another street address. For example, a brochure for a local bowling alley states that they are "opposite the cinema". This additional locative information is intended to assist the requestor of the service. This proximity is captured in Figure 3.18 using a preposition. The enumeration constraints that we present in the model (such as opposite, next, above, below) are not intended to be complete but rather indicative of the type of information that the Preposition entity captures.

Postbox addresses we refer to as having either a general box number, a private box number or a locked bag number. They also inherit the same properties of the Address supertype that were mentioned above.

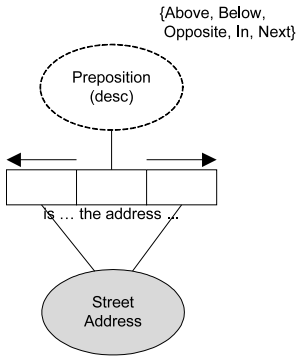


Figure 3.18: Proximity of standard addresses

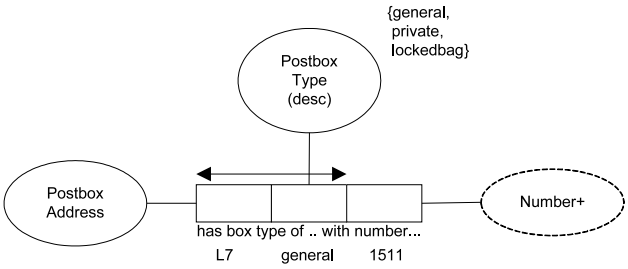


Figure 3.19: Postal box addresses

3.2.5 Street directory reference

A street directory reference is a common way for service providers to provide assistance with physically finding a place of service provision [see Figure 3.20]. Street directories are commonly published books that include the following properties: an ISBN code (that uniquely identifies the book), the book title, the provider, the edition (e.g. 3rd edition), the region that the map relates to (e.g. Sunshine Coast or Gold Coast), the map number, the x and y position within the map, and the publication date. For example, Saint Joseph's college in Brisbane can be identified using the following street directory reference - UBD Map 120 Ref 2N.

The ReferencePosition entity depicted in the model has indicative enumeration constraints of 1 to 1000, and A through Z. This entity is used to reflect the X and Y position on the street directory. The values of the enumeration constraints are intended to be illustrative only.

3.2.6 Phone numbers

Phone numbers are commonly included in the service descriptions as a means of requesting the service. We distinguish between fixed and mobile/cell phone types [see Figure 3.21]. All phone numbers are considered to be functionally constrained to support one or more of the following interaction types: voice, modem, SMS text messaging, facsimile and telex communications. We consider that it may be necessary to capture the international direct dialling prefix, the country code, the national direct dialling prefix, and the city/area code. It is mandatory to store the

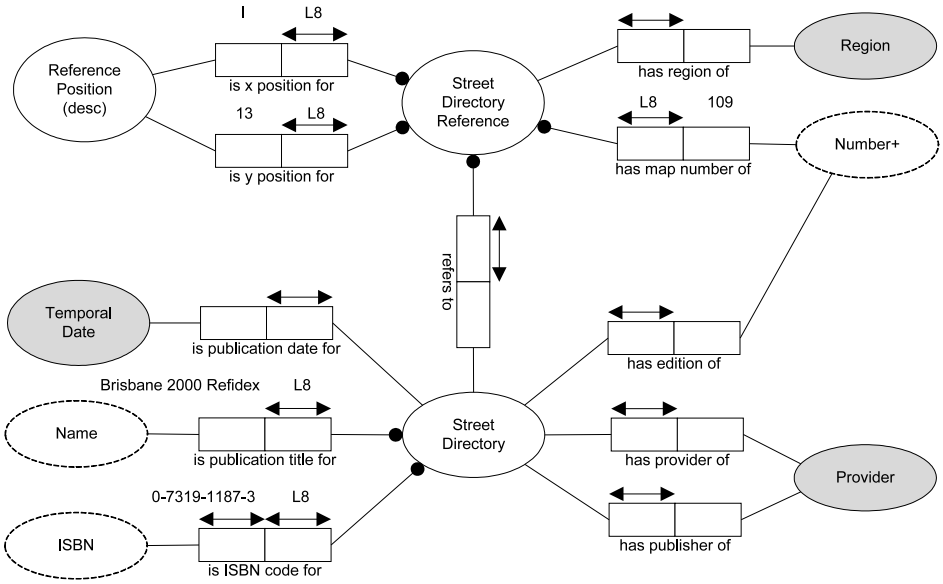


Figure 3.20: Street directories

local phone number.

The discovery of information in this model is dependent on where you are calling from and where you are calling to. If you are located in Victoria (Australia) and you are calling someone in the state of Queensland then you need to ring the national direct dialling prefix "0", the area code "7" and the eight digit local number. If you are located overseas in Germany (and you wish to ring the same Australian local number) then you would need to ring the international direct dialling prefix "00", the country code "61", the area code "7" and the eight digit local number. The international direct dialling prefix will be dependent on the region that you are calling from, and possibly the telecommunications carrier that you are using to make the call. We support the notion of toll free (or free call) phone numbers for callers from certain regions.

The following example ConQuer query filters phone number instances to be those of "FixedLine" type that have a country code of "61" (being Australia) and an area code of "7" (for Queensland).

PhoneNumber
- ⊢ has phone number type "FixedLine"
- ⊢ has country code 61
- ⊢ has an area code 7

3.2.7 Uniform resource identifiers

Uniform Resource Identifiers (URI) are used to describe abstract or physical resources [9]. They consist of a scheme type (e.g. ftp, http, mailto, news, gopher,

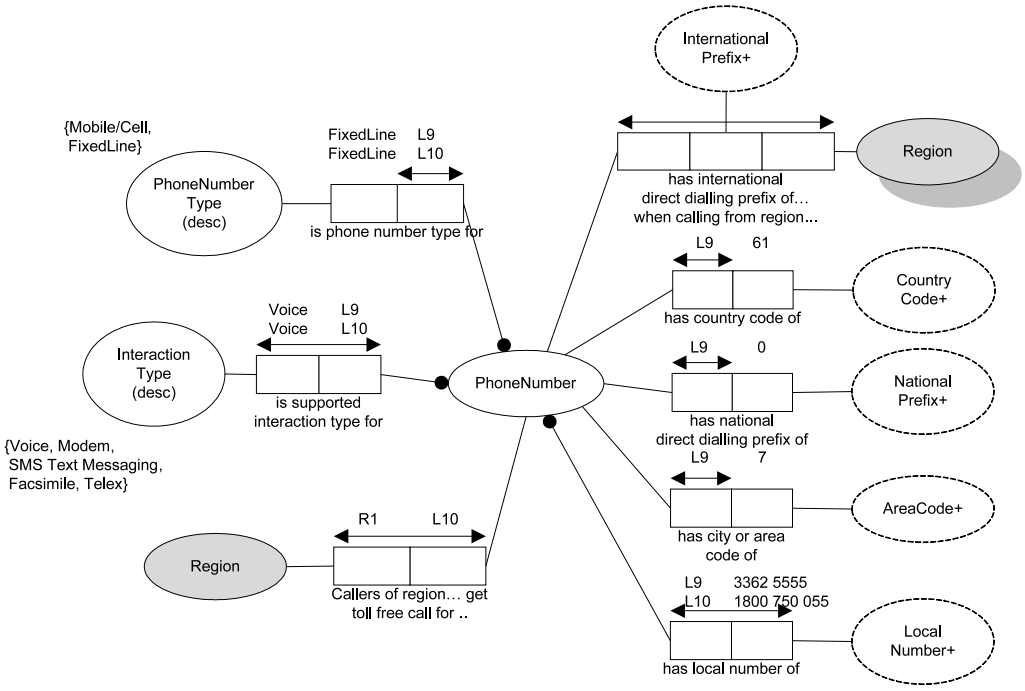


Figure 3.21: Phone numbers

telnet), an authority, a path and possibly a query [see Figure 3.22]. An authority represents the domain of the server. It includes a server name, userinfo and a password. Although the latter two are not recommended for use due to security concerns, we include them here for completeness. We consider web services to be capable of using URIs as a means of identifying the endpoint that facilitates the service provision.

The following example query filters for instances of URIs that have a scheme type of "http" and where the authority server name contains the word "google". The use of the contains function within the query enables us to get authorities such as gmail.google.com, labs.google.com, www.google.com and www.google.com.au.

URI
- ⊢ has scheme type of "http"
- ⊢ is Authority
  - ⊢ has server name like "%google%"

3.2.8 Internet Protocol addresses

The Internet Protocol (IP) was designed to provide functionality for non-reliable, and non-sequential delivery of data packets (i.e. a collection of bits also known as a datagram) from a source address to a destination address. The transfer of these packets may involve many interconnected networks [32].

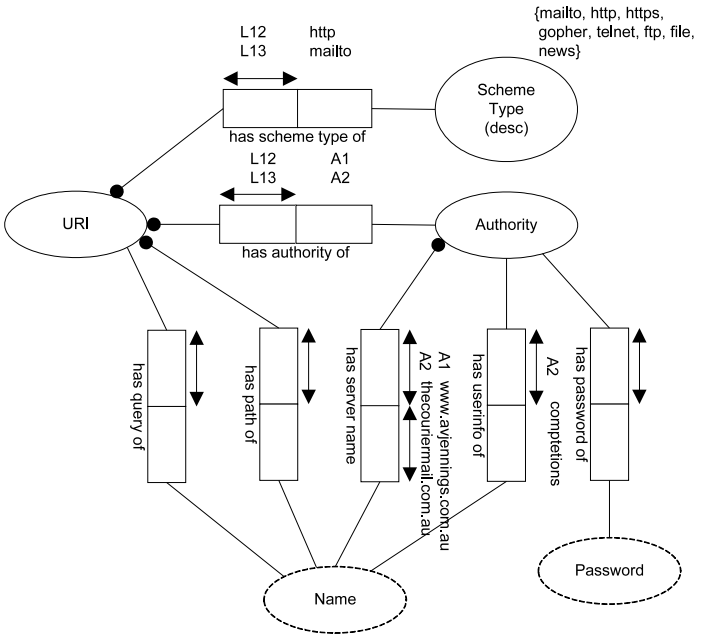


Figure 3.22: Uniform resource identifiers

IP addresses are 32-bit, four part addresses used to define the network and host connected to an Internet Protocol network [see Figure 3.23]. IP addresses have a class A, B, C and local address components. Each part is referred to using a number between 0 and 255. The four parts are concatenated and are represented in the form A.B.C.local (e.g. 131.181.118.220, an address registered to the Queensland University of Technology). Whilst a relationship exists between the IP address and a URI authority server name we believe that service providers will utilise one technique for description, not both.

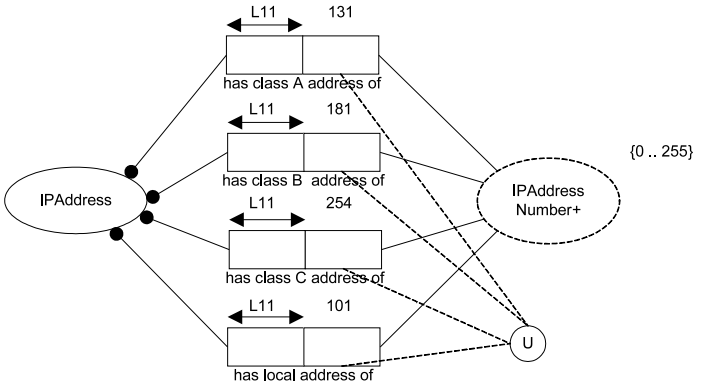


Figure 3.23: Internet Protocol addresses

We do not currently model IPv6 addresses.

3.2.9 Ethernet addresses

The Address Resolution Protocol (ARP) is used to map IP addresses to hardware addresses [79]. This protocol can be considered a form of service. Ethernet addresses are used to uniquely describe a computer connected to an Ethernet network [see Figure 3.24] [28]. Ethernet addresses consist of 48 bits. They consist of two parts, the first is an identifier for the Ethernet hardware provider (also called an Organizationally Unique Identifier or OUI), the second is the address of the computer on the network. They are commonly written in hexadecimal format and are used to uniquely identify an ethernet networked device. Ethernet addresses are sometimes referred to as physical, hardware or MAC addresses.

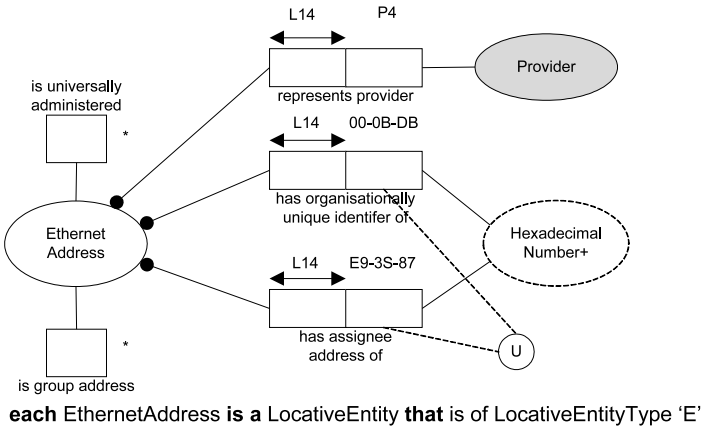


Figure 3.24: Ethernet addresses

As the IEEE assigns the OUI, this effectively reserves a range of addresses for that provider. Whilst both parts of the address are 24 bits in length the true length of the first part is 22 bits. The other two bits denote a:

- Individual or Group address - Identifies an address as an individual or group address; and a
- Universally or locally administered address - Identifies whether the address has been assigned by a universal or local administrator.

Both of these can be derived by looking at the OUI. Using the individual or group bit provides for approximately 32 million (16 million local and 16 million group) addresses. These derived fact types are denoted in Figure 3.24 with an asterisk.

3.2.10 Spectra

Spectra is the collective term that we use to describe electromagnetic waves (or radio waves) for AM radio, FM radio, citizen's band (or CB) radio, TV, microwave, infrared, short wave, and radio frequency identification (RFID) that operate within specific frequency bands [see Figure 3.25]. For each spectrum we capture the region

within which the spectrum is available. This caters for variations between countries for each type of spectrum.

We consider each spectrum to operate within a frequency band. A frequency band is defined by lower and upper boundaries for a specific band (e.g. AM radio normally operates in the 535 kilohertz to 1.7 megahertz range). When the lower and upper frequency are defined, they include a frequency number (e.g. 535) and the oscillation frequency units (e.g. kilohertz). The oscillation frequency refers to the number of cycles per second that the radio wave oscillates at. Therefore 535 kilohertz is 535 thousand cycles per second. It is not within the scope of this paper to present a full list of frequency bands by region.

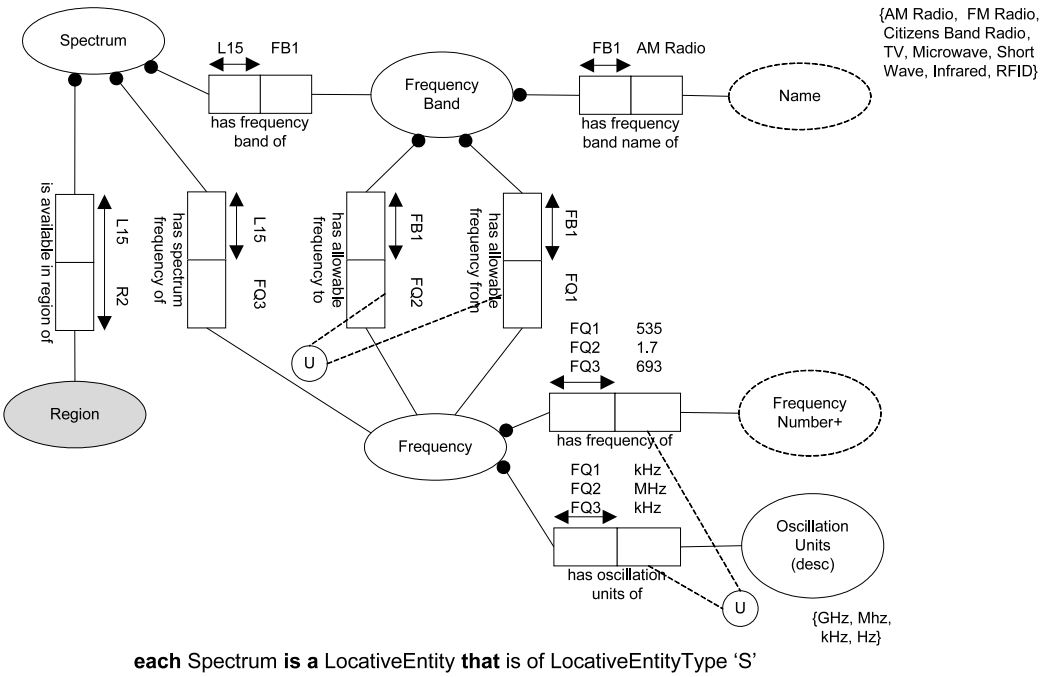


Figure 3.25: Spectra addresses

RFID tags can operate in either low, high, UHF or microwave frequencies [115]. We do not presently consider the fact that the RFID tag is either active or passive (referring to the manner in which the tag is powered), or the fact that it is assigned a globally unique identifier to be of high relevance for capturing the locative availability of a service.

Whilst other spectra exist for technologies such as garage door openers, baby monitors, traffic control systems and global positioning systems, we have tried to provide for the primary frequency bands that are in use. We do recognise that mobile phones operate using radio waves (typically in the 824 to 849 Mhz range) but chose to model them with phone numbers (see section 3.2.6).

With foundations for describing temporal and locative entities we are now able to utilise these in a number of different contexts, including description of the availability of a service. We have offered a semantically rich representation for both the *where* and *when* aspects of a service.

3.3 Service availability

Service availability refers to the combined use of temporal and locative aspects of services to describe when they can be requested, provided, approached for issue resolution, approached with feedback, or queried for more information. We have split our modelling of service availability into two parts (based on the work of Piccinelli [77]) - availability related to the requests for services [see Figure 3.26] and availability related to provision [see Figure 3.27]. We deal with the former first.

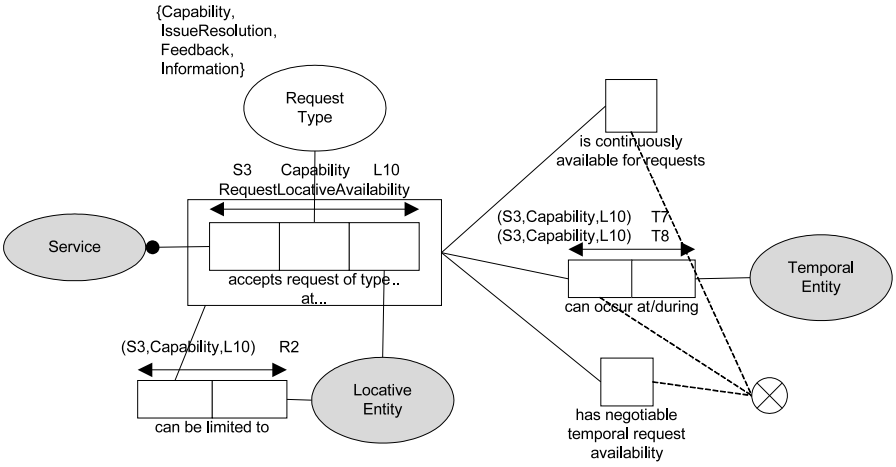


Figure 3.26: Service request availability

Requests to providers can be for the service capability, to undertake issue resolution, to provide feedback or to gather information from the service provider. We consider the combination of the service, the request type (i.e. capability, issue resolution, feedback, information) and the locative entity to be the request locative availability [see the objectified type in Figure 3.26]. We apply the ability to filter (i.e. limit) this request locative availability to ensure that requests are from specific locations. This is useful in the context of franchises that service locations such as regions (or a group of suburbs). Having shown where service requests can be conveyed to, we now attach the temporal availability of the request. In doing so, we complete the request availability from both a locative and temporal perspective. We consider the requests to have three declared types of temporal availability (depicted on the right hand side of Figure 3.26):

- Nominated availability - The service requests are accepted at/during a specific temporal pattern (stated using a temporal entity).

- Negotiable availability - The service requests can be configured by the requestor in conjunction by discussing temporal availability with the provider.

- Continuous availability - The service is continuously available for requests (e.g. for use in the context of web services).

Alternatively, a provider may elect not to advertise the temporal availability of requests for their service(s). This then communicates to service requestors only the locative availability for requests to a service.

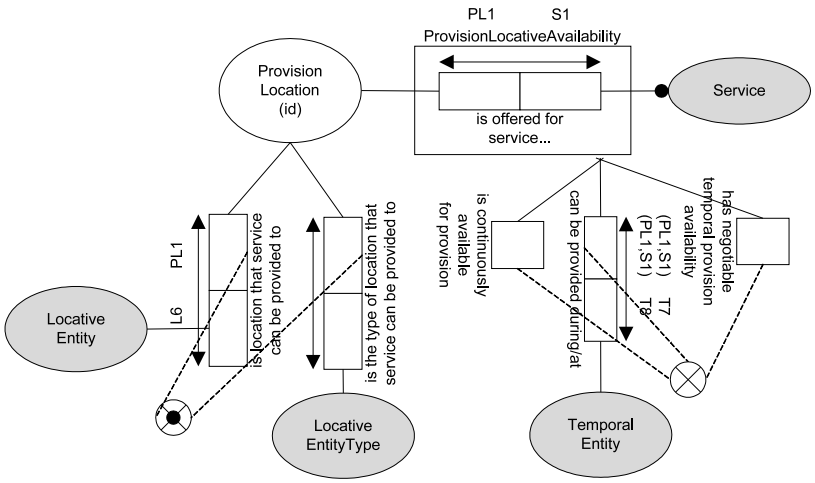


Figure 3.27: Service provision availability

The provision side of availability is somewhat different as it offers the ability to describe either a specific locative entity that is "where" a service can be delivered to, or a type of locative entity (e.g. an address, a region, a URI etc). This is useful as a service provider may not want to provide a complete list of addresses that they are willing to provide a service to. However, it is important to note that since the locative side of provision can be expressed in either of these two ways, the equivalent objectification of the locative fact type requires the introduction of the ProvisionLocation entity. We then allow for the attaching of three types of declared temporal availability with respect to provision: a nominated availability (i.e. a specific temporal entity is provided), a negotiable availability or continuous availability. Alternatively, a provider may elect not to advertise the temporal availability for provision of their service(s).

3.4 Communication

We consider communication between the service requestor and service provider to be intrinsically linked to the method of request or provision (e.g. email, web site, web service). To this end we offer three types of support for communication within our model [see Figure 3.28]. We attach this support to the locative entity which the service requestor interacts with. The three types of communication support are:

- That a service is capable of interacting with a service requestor in a written manner using a language defined within the ISO639-2 standard [97]. ISO639-2 uses a three letter character code to represent a language, or a family of languages.

- That a service is capable of interacting with a service requestor in a verbal manner using a language defined within ISO639-2; or

- Finally, that a service is capable of interacting with a service requestor according to a standard (e.g. Web Services Description Language - WSDL). We provide a discussion of the entity type "Standard" in section 6.2.

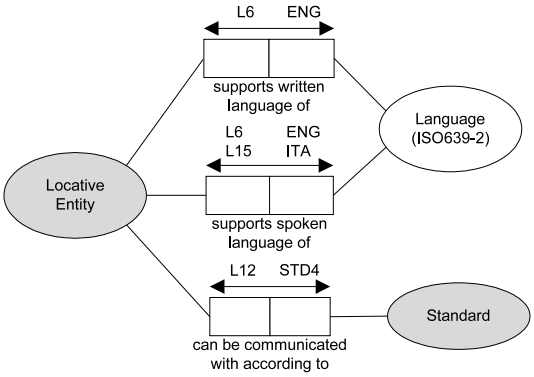


Figure 3.28: Communication

Examples of ISO639-2 codes include: "afr" for Afrikaans, "eng" for English, "fin" for Finnish and "ita" for Italian. In some instances, ISO639-2 uses a combination of two three-letter codes to describe the language. The first refers to a bibliographic code (sometimes termed ISO639-2/B) and the latter refers to a terminology code (sometimes referred to as ISO639-2/T). We choose not to distinguish between the type of code (i.e. bibliographic or terminology) and provide the capability to store multiple ISO639-2 codes against a single locative entity.

Our final type of support, that treats locations as being capable of communicating according to a standard, whilst appearing cursory caters for the storage of multiple standards against a single location. This is useful when two or more types of standards are necessary for description of the service. For example, WSDL describes the operations and messages that a web service exposes. We could additionally attach a reference to the choreography of messages that cannot be captured using WSDL. This choreographical reference could be stated as an instance of a particular standard.

Summary

This chapter has presented a discussion of service availability. Fundamental to our ability to describe availability are our temporal and locative concepts. We have presented a range of semantically rich models with respect to temporal and locative entities. These models are subsequently used to describe service availability. We have also offered a discussion around communication which we associate with locative entities.

The next chapter is a discussion of obligations, payment and price. These same temporal and locative entities are used regularly within these models to capture aspects such as when payment is due, where payment can be made etc.

Chapter 4

Obligations, Price and Payment

This chapter is a discussion of obligations, price and payment. We use the term *obligation* to capture the notion of a duty or requirement on a party to the service request or provision. We outline the different obligations that are compelled upon the service requestor or the service provider. These include obligations such as a relationship obligation, where a service requestor is expected to be bound to the service provider for a specified period.

We present these three categories of non-functional properties together for the following reason. After temporal and locative availability, price and payment were commonly found together during our analysis of existing service descriptions. Price and payment are complementary, and we find the notion of obligation to be an abstraction that both binds them together, but also acts as an extension point for domain specific obligations.

The non-functional properties of price and payment are quite complex in their nature. This is due to the various factors such as location of the requestor, context of requestor (e.g. their age), what payment mechanisms the service provider will allow. We explore these factors in detail in sections 4.2 and 4.3. Finally we explore the concept of reward schemes and how they relate to both payment and price.

4.1 Obligations

In order to capture the responsibilities of both the service provider and the service requestor we offer the notion of "obligations" as a means of ensuring that these non-functional properties are available for discovery by interested parties. We believe that by recording obligations, the discovery process will be greatly enhanced.

Obligations can be attached to either the request for a service, or the provision of a service [see Figure 4.1]. For example, a service provider may wish to advertise that service requestors have an obligation for a relationship, or an obligation to make payment should they request their service. It is the service provider who must fulfil the obligations relating to the service provision. As one or more providers may be involved in the provision of a service, we represent the obligations of the service provider separately to service requestors. We don't associate obligations with individual service requestors, as it is unreasonable to expect a service provider

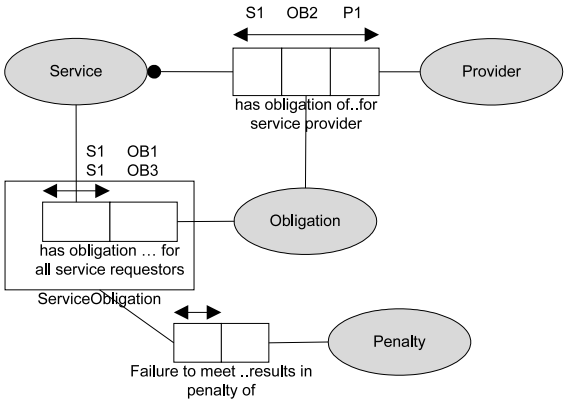
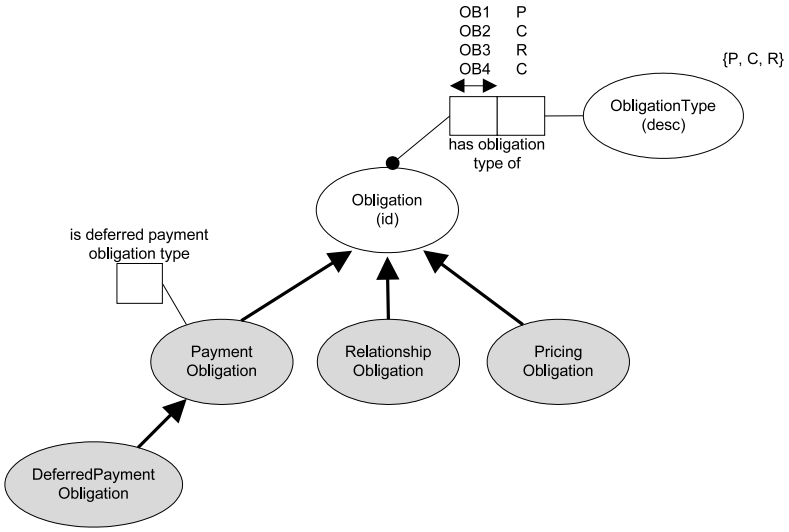


Figure 4.1: Service obligations

to outline all instances of service requestors to whom the obligations apply. Our model allows the obligations of multiple service providers to be included within the description. This may be necessary under the scenario of a service composition where each provider has distinct obligations. The nested entity type of ServiceObligation allows us to attach penalties to the failure to meet these obligations. Penalties are discussed in more depth in Chapter 5.

We now discuss three obligations: pricing, payment, and relationships [see Figure 4.2]. In future, other obligations may be added to further increase the expressiveness of service descriptions.



each PaymentObligation **is a** Obligation **that** is of ObligationType ‘P’
each RelationshipObligation **is a** Obligation **that** is of ObligationType ‘R’
each PricingObligation **is a** Obligation **that** is of ObligationType ‘C’
each DeferredPaymentObligation **is a** PaymentObligation **that** is deferred payment obligation type

Figure 4.2: Obligations

4.1.1 Pricing obligations

We view the need to price a service as an obligation of the service provider [see Figure 4.3]. The pricing obligation can be considered to wrap the price of a service with many other important non-functional properties. Most pricing obligations will primarily be concerned with capturing the price, but it is also necessary to capture information such as:

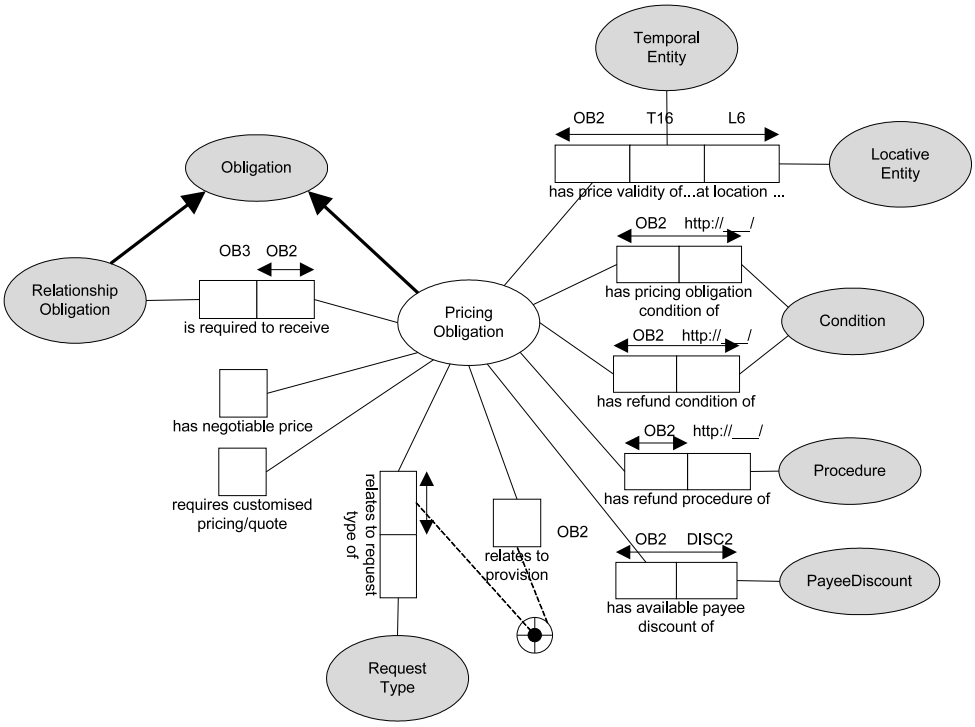


Figure 4.3: Pricing obligations

- Conditions - these relate to any specific requirements or restrictions to the price, or to the refund for the price paid for a service. Conditions are complex entities that we identify according to a URL. The constraints over the condition role allows for multiple URLs to be specified. This provides an ability to define fine grained conditions for use by service providers. Price conditions may include limiting service requestors to a particular group (e.g. elderly) or to the exclusion of a particular group (e.g. requestors of government agencies). We also attach the ability to define specific refund conditions that are common for transportation services such as plane tickets where they state that a ticket may not be refundable, or may only be refunded within a particular timeframe. A discussion of our treatment of conditions is presented in section 1.5.

- Refund procedures - associated with the specifying of refund conditions it may also be necessary for a service provider to state a refund procedure. This procedure is used by service requestors to enact the refund process. We consider procedures to be a sequence of steps that are followed to achieve an outcome.

Our treatment of procedures is as per conditions. We utilise a URL reference to provide a link to the description of the procedure.

- Price validity - this provides a where and when scoping of the price's availability. Using our temporal models the temporal validity can be specified as an anchored or recurring interval, an anchored temporal instant or a date. We also capture the location, as the pricing obligation may be specific to a limited number of the locations.

- Negotiability - sometimes service providers may advertise a price but be willing to accept a lesser amount. Our model allows the provider to state that they are willing to negotiate on price.

- Price customisation - this allows providers to capture that their service is highly customisable, and therefore the actual price cannot be expressed (e.g. a landscaping service may not be able to express the price until they have an understanding of the requestor's block of land and their objectives). This does not reduce the usefulness of the service description as the service provider is still capable of expressing the other pricing obligation related properties within this list, as well as other non-functional properties.

- Relationship obligations - this allows service providers to state that a relationship is required before they will commit to a price and its surrounding non-functional properties (e.g. quality, security, rights). See the outline of relationship obligations provided in the latter part of this section. It is possible to specify a relationship obligation within the model presented in Figure 4.1 that refers to the need to have a relationship with the service provider to receive the service output. We can also state a relationship obligation for the service requestor that is attached to a specific pricing obligation. For services that specify multiple pricing obligations (assumed to state different prices), then differing relationship obligations for those prices could be described.

- Payee discounts - we provide an in-depth discussion of discounts in section 5.1 but provide a link within our pricing obligation model to one specific type of discount, those related to who the payee is. This might include a person from a particular age group (e.g. the elderly), those with membership to a particular body, or even a shareholder of a company.

As stated in our availability models, we view availability from the request and the provision perspective. Finally, we attach the price to either the type of request or to the provision. We leave it up to service providers if they would like to state their price based on one of the request types (capability, issue resolution, feedback or information), or provision.

Deliberately, the actual price (e.g. $20 USD) for the service is not included within this model. We present a detailed discussion in section 4.2. This section was intended to be a discussion of the notion of a pricing obligation.

4.1.2 Payment obligations

The payment obligation is the service requestor's obligation to pay the service provider. The payment obligation primarily provides a set of links to one or more pricing obligations. We have defined fact types within our payment obligation model [see Figure 4.4] for the following:

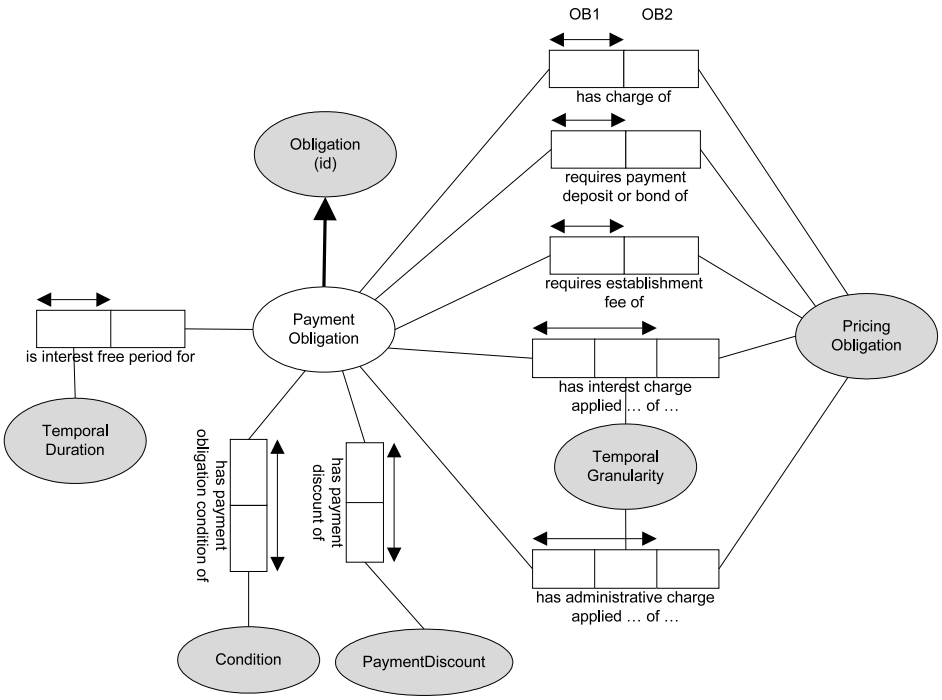


Figure 4.4: Payment obligations

- Charges - this is the relationship from the payment obligation to the primary pricing obligation (i.e. the price of the service). We don't associate it directly with the price of a service as it is important to highlight the price of the service, as well as the other pricing related non-functional properties that we presented in section 4.1.1.

- Establishment fees - this allows the service provider to state that there is an obligation to pay a once-off fee for first time use of a service. Establishment fees (sometimes called "joining" fees) are common with gymnasiums and professional bodies.

- Interest charges - not specified by all services, interest charges are considered payable on services such as mortgage loans. If an interest charge is specified, it may be as an alternative to stating the charge of a service. An interest free period may be stated as a temporal duration.

- Administrative charges - this type of charge is common across many services. Examples include: (a) banks who charge monthly fees for some transaction

accounts, and annual package fees for some mortgages; and (b) rental property managers who charge monthly postage and petty expenses costs. An administrative charge is normally stated with a frequency (e.g. annual, weekly, monthly, daily etc).

- Deposits/Bonds - service providers can choose to outline the need to provide a deposit or bond that accompanies a service. Deposits are common for a range of services including when people rent a property, or when borrowing hire equipment. This is another example of the usefulness of attaching the payment obligation to the pricing obligation, rather than the price. In doing so, we can then navigate to the refund conditions and procedures for the deposit/bond.
- Payment discounts - we provide a link within our payment obligation model to one specific type of discount, those related to how you pay. For example, this includes discounts related to the use of certain payment instruments, and payments to a certain type of location.
- Conditions - we allow for the attachment of conditions to the payment obligation.

Deliberately, the manner in which a requestor can go about making payment (e.g. using a credit card) to the service provider is not included within this model. A detailed discussion of this and the properties of payment instruments is provided in section 4.3. This section was intended to be a discussion of the notion of a payment obligation.

Deferred payment obligations

A subtype of payment obligation is "DeferredPaymentObligation". These are a specialisation of a payment obligation [see Figure 4.5] that enable a service provider to accept deferred payment for a service over a specific period (possibly after provision of the service has completed). Our deferred payment model includes support for specific deferred payment conditions, a deferred payment period stated as a temporal duration (e.g. one year), one or more temporal entities that represent a schedule of deferred payments, and the minimum amount that must be spent by the service requestor on the service to enable them to get the deferred payment option. All deferred payment obligations also inherit the properties of the supertype (i.e. PaymentObligation). Deferred payment obligations are common for retailers selling furniture, whitegoods, and entertainment systems. For example, large retailers may offer an 18 month (or more) interest free period for purchases over a certain amount (e.g. $500).

4.1.3 Relationship obligations

We provide the ability to capture an obligation that is a mandatory commitment by the service requestor to the service provider for a specific period [see Figure 4.6].

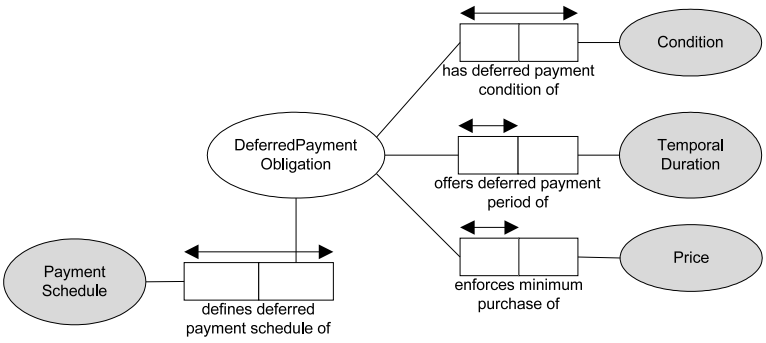


Figure 4.5: Deferred payment obligations

We refer to this as a relationship obligation. Relationship obligations include a set of conditions that govern the relationship, a minimum relationship duration (e.g. 12 months), and an associated set of rights. We allow a service provider to attach a set of rights that are associated with the relationship obligation. For example, this may be a service provider's right to disclosure, or a service requestor's right to privacy or recourse.

We have a specific subtype of relationship obligation that we call "Membership". Memberships are a common term used to describe an extended commitment. Memberships usually include a set of membership levels. We optionally attach a price to membership levels.

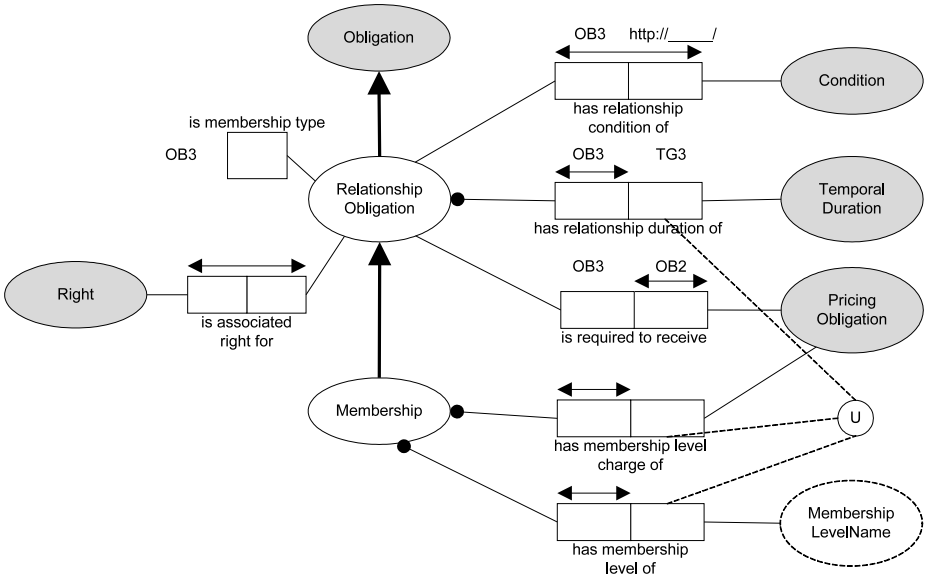


Figure 4.6: Relationship obligations

It is important to note that whilst each of the obligations outlined above has a specific fact type for capturing conditions, it is possible to have represented this as a single fact type associated with the Obligation supertype. We have chosen

specifically to attach the condition to each subtype to retain further semantics about what the conditions relate to (i.e. deferred payment, relationship commitment).

4.2 Price

As previously noted, we perceive the non-functional properties of price and payment as complementary [71]. We interchangeably refer to price as cost: cost being the view from a service requestor perspective, whilst price is the view from the service provider perspective. Within this section we refer to price as the amount being charged for a service. We believe that the pricing of a service is an obligation of the service provider.

Examples of price descriptions include:

- Carpet cleaning service - a carpet dry cleaning service offers three rooms cleaned for $89 AUD (where the maximum room size is 13 sq m, and subject to inspection of the carpet condition). They also offer two rooms for $69 AUD with additional rooms $25 AUD per room. Four rooms cost $110 AUD.
- Newspaper delivery service - a newspaper offers home delivery of newspapers daily for $7.20 AUD per week (i.e. seven days for $7.20 AUD).
- Accommodation service - a hotel in Surfers Paradise (Australia) offers a room for $82.50 AUD per adult twin share.
- Mobile phone service - a phone carrier provides a mobile phone service. It is $10.38 AUD per month for 24 months for the handset, whilst the associated plan costs $25 AUD per month.
- Property investment syndicate - this investment vehicle offers 4.0% per annum income paid each six months. There are no fees.

From these examples we can see that prices are complex entities. They are not always easily captured as a simple dollar value in a certain currency. Prices do become domain specific when the quantifiers (e.g. per room) are applied. Certain complex conditions may also surround the ability of a service requestor to receive the advertised price.

We consider that the pricing obligation of the service provider, in conjunction with the price, produces a new entity that we refer to as the "ServicePrice". We attach further information to this entity later in this section (e.g. tax, price granularity, price modifier). We also consider that after stating a price (e.g. ten nights at $150 USD per night) the service provider might like to attach the price for additional invocations (e.g. each extra night is $100 USD per night). We assert that every price is one of the following [see Figure 4.7]:

- Absolute prices - these contain a specific amount and a currency. For example $10 AUD represents ten (10) Australian dollars. We specify currencies according to the International Organization for Standardization (ISO) standard - *ISO 4217:2001 Codes for the representation of currencies and funds* [50].

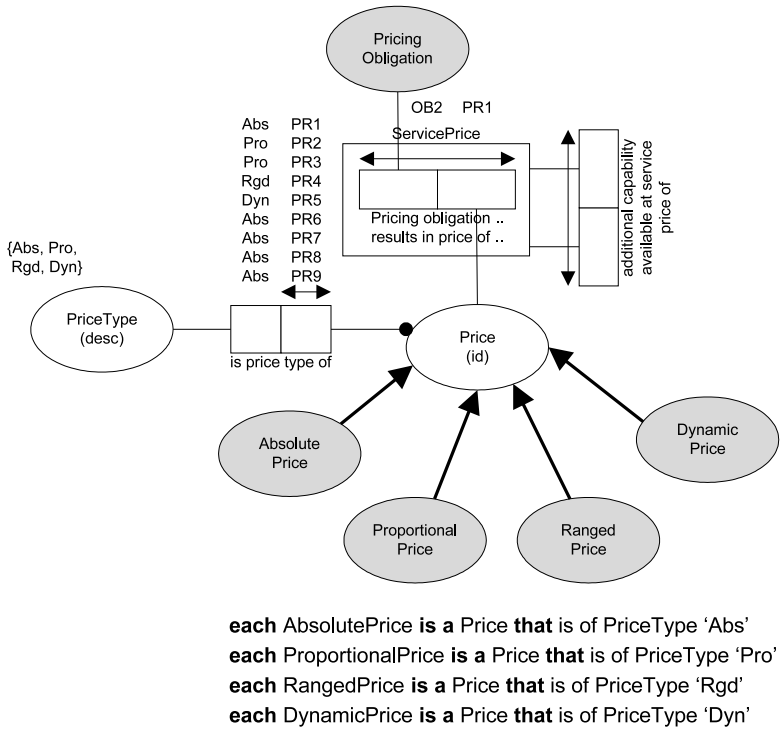


Figure 4.7: Price

- Proportional prices - these represent a percentage value with respect to a certain item. We refer to those items using URIs. This allows us to point to external ontologies or dictionaries that provide definitions. For example, the price of entering a managed fund might be 2.5% of the value being invested into the fund. The URI that defines "the value being invested in the fund" might be "`http://www.service-description.com/finance.htm#Funds_under_mgmnt`".

- Ranged prices - these are further subdivided into one of two types [see Figure 4.12]:

  - Ranged absolute - a ranged absolute price contains a *from* and *to* value that are both AbsolutePrice entities. For example, a service provider may prefer to provide a ranged price rather than a specific price (e.g. $150,000 USD - $175,000 USD). This means that the service provider may achieve a higher price if a service requestor is not familiar with the current market value of a service. In other cases, ranged prices reflect that various options are available that differ the final price of the service, depending on the option(s) selected.

  - Ranged proportional - a ranged proportional price contains a *from* and *to* value that are both ProportionalPrice entities. For example, a service provider may state the cost of their service as a range between 1.5% and 3% of the final sale price.

- Dynamic prices - this form of pricing captures mechanisms like auctions, where the price is determined by a market's natural supply and demand. We capture the type of mechanism (such as English auction, Dutch auction etc), the conditions associated with using the mechanism, the location and temporal availability of the mechanism, and a reserve price (as either an absolute or proportional price) [see Figure 4.13].

Price also includes an item granularity that is applicable to all types of prices [see Figure 4.8]. The item granularity reflects a general granularity, a cardinality and an item granularity number that provides ordering of granularities. To ensure that the ordering of item granularity numbers is sequential, an additional rule (depicted using a circled "R") is included within the model. The granularity of the item in turn refers to a unit of measure. We foresee the use of common granularities such as those presented in Table 4.1. These granularities could be extended further to support notions such as a room. This caters for services such as the carpet dry cleaning example that was presented at the start of this section.

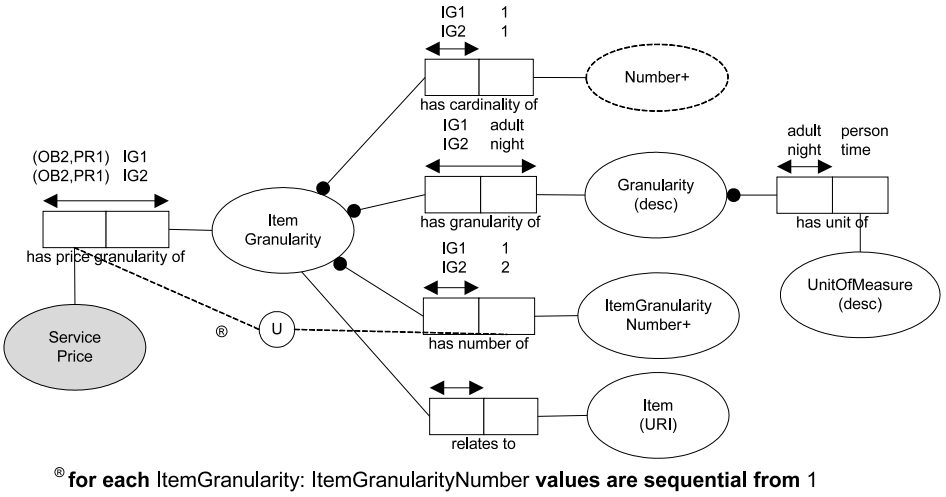


Figure 4.8: Price Granularity

The following example query filters instances of price that are stated as an hourly rate between $10 and $15 United States Dollars (USD). This should not be confused with ranged prices where a service provider states the price using two absolute prices.

AbsolutePrice
- ⊢ has amount >= $10.00
- ⊢ has amount <= $15.00
- ⊢ has currency of "USD"
- ⊢ has price granularity of ItemGranularity
  - ⊢ has granularity "Hour"
  - ⊢ has unit of measure "Time"
  - ⊢ has cardinality 1

$\vdash$ has item granularity number 1

Unit	Granularity
Time	Hour, Minute, Second, Day, Month, Year, Night, Week, Fortnight.
Weight	Gram, Kilogram, Tonne.
Volume	Cubic metre.
Area	Metres squared, Square metres.
Length	Millimetre, Centimetre, Metre, Kilometre.
Watt	Kilowatt, Megawatt.
Byte	Kilobyte, Megabyte, Gigabyte.
Person	Adult, Child, Infant, Pensioner, Senior.
Event	Mouse click.
Permit	Ticket.

Table 4.1: Price granularity

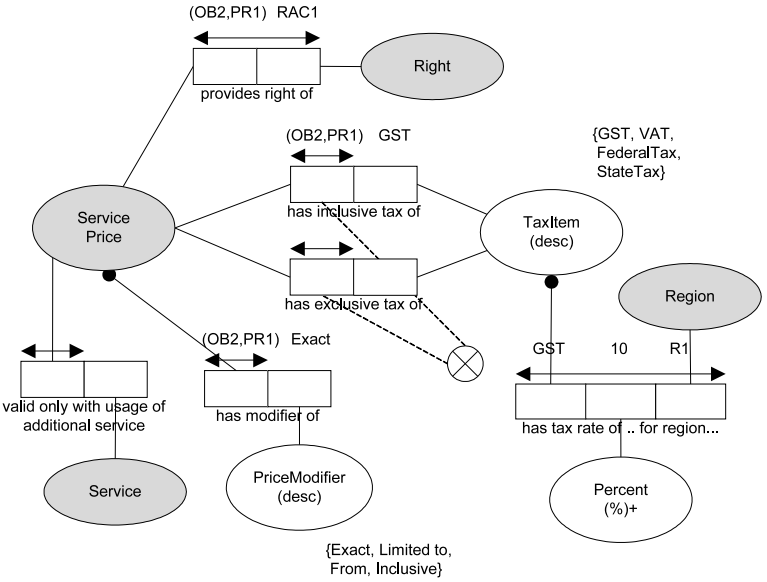


Figure 4.9: Pricing - Tax and modifiers

All prices have a modifier that quantifies the price being specified [see Figure 4.9]. We suggest four possible modifiers: exact, limited to (the price will not go higher than the amount specified), inclusive (intended for ranges of values) and from (the price starts at this amount and will go higher depending on how the service is configured by the requestor). Prices may include a component that is tax related [see Figure 4.9]. Service providers can choose to state their price as inclusive or exclusive of a tax item. If a tax item is captured, then a tax percentage is attached. For example, Australians are taxed at a rate of 10% on the majority of goods and services they purchase under the Goods and Services Tax (GST). Similar types of

taxes include the Value Added Tax (VAT). Tax is applicable to a particular region. Some services offer a price based on the criterion that the service requestor also makes use of another service. An example is where the carpet cleaning service provider will offer its carpet protection service only when addition cleaning services are purchased. A service price may also provide either the service requestor or the service provider with one or more rights with respect to the service. Rights are outlined in more depth in section 5.3.

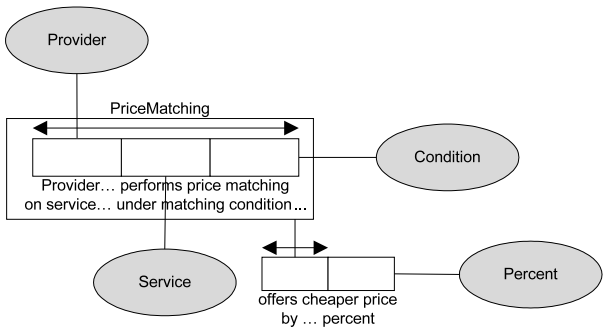


Figure 4.10: Pricing Matching

We provide a price matching facility within our price model [see Figure 4.10]. Some services advertise that they are willing to match or better the price of another competitor. For this type of service provider we allow the attachment of a percentage which indicates what they are willing to improve competitor offers by (e.g. 5%).

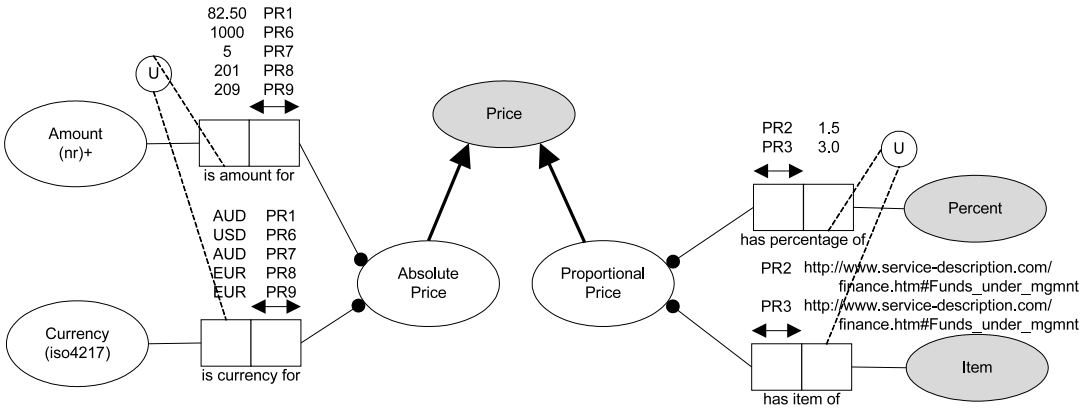


Figure 4.11: Absolute and proportional pricing

The following example query filters instances of price based on their annual, tax exclusive value where the proportional price ranges from 1.5% to 3.0% of "Funds under management". We refer to the item being described using a URI (e.g. `http://www.service-description.com/finance.htm#Funds_under_mgmnt`).

RangedProportional
⊢ has exclusive tax "VAT"

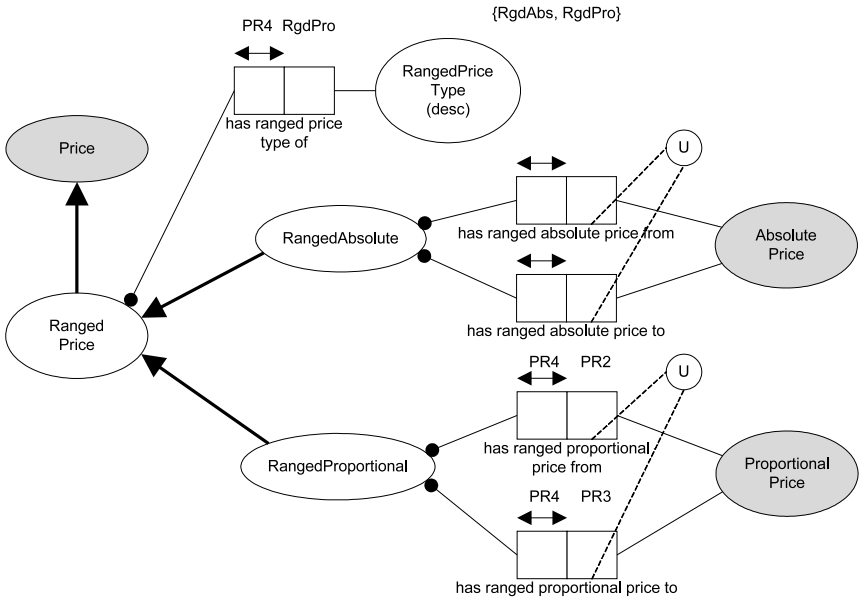


Figure 4.12: Ranged pricing

⊢ has price granularity of ItemGranularity
  ⊢ has granularity "Year"
  ⊢ has unit of measure "Time"
  ⊢ has cardinality 1
  ⊢ has item granularity number 1
⊢ has ranged proportional price of ProportionalPrice
  ⊢ has percentage 1.5%
  ⊢ has item "http://www.service-description.com/finance.htm#Funds_under_mgmnt"
⊢ has ranged proportional price of ProportionalPrice
  ⊢ has percentage 3.0%
  ⊢ has item "http://www.service-description.com/finance.htm#Funds_under_mgmnt"

Dynamic pricing mechanisms largely refer to online auction sites such as eBay where the market determines the price for a particular item [see Figure 4.13]. These mechanisms normally provide a temporal window at a particular location (e.g. a URL) where the mechanism is available. We use the TemporalInterval entity to allow for anchored intervals and recurring intervals. Auctions therefore can be described as occurring at a regular temporal interval (e.g. every Monday at 9am). Providers who are auctioning their services can state a reserve price and/or an indicative price. There is normally an array of specific conditions associated with these mechanisms. We do not attempt to capture these conditions. A mechanism type is attached to the dynamic price. This refers to dynamic pricing mechanisms such as Dutch

auction, English auction, continuous double auction, Vickrey auction, sealed bids and request for tender. We provide a link to the provider of the dynamic pricing mechanism.

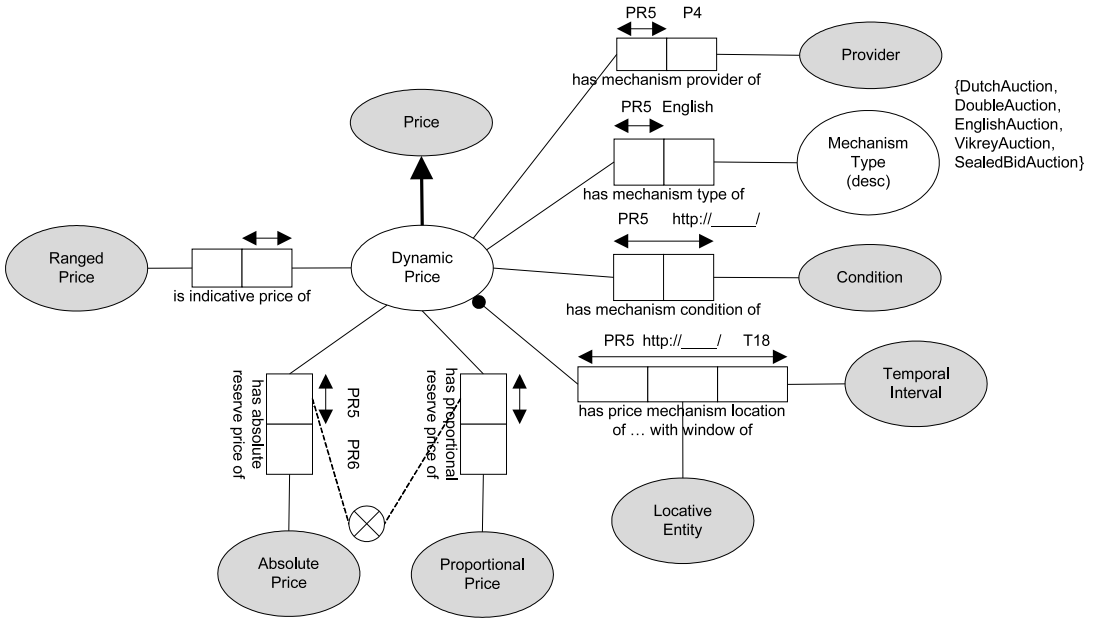


Figure 4.13: Dynamic pricing

4.3 Payment

In this section we present models for payment options, payment schedules, payment instruments and instrument types. We view payment as complementary to price. It captures the manner in which a service requestor can fulfil their payment obligation. Consider the following examples of payment related descriptions:

- A restaurant has dining, catering and cooking school related capabilities. The service provider expects payment to be due on the invoicing date, or has 7 day terms for prior arrangement. It prefers the use of either direct deposit or business cheque. Payment with an American Express or Diners card attracts a surcharge of 5%, whilst Visa, MasterCard and Bankcard attract a 2% surcharge. They offer a 5% discount for cash payment on functions over $1000 in value.

- In an online auction service, the seller of a particular item offers payment options based on the purchaser's location. For requestors inside Australia the following are available: direct deposit/transfer, cheques, money orders, cash, PayPal or Paymate. For requestors outside Australia the following are available: International money orders, and Cash (Pounds sterling, Australian or United States dollars).

- An amusement park accepts Australian traveller's cheques, Bankcard, cash, bank cheques, Diners Club card, EFTPOS, foreign travellor's cheques, Mastercard and Visa.
- An electrician accepts EFTPOS, Mastercard, Visa, Bankcard, personal or bank cheques and cash.

In simple terms we view a payment option as the preparedness of a service provider to accept a particular payment instrument(s) at one of several payment location(s) [see Figure 4.14]. The relationship between the payment obligation and a payment option is referred to as the "ServicePayment". The service payment primarily associates the use of certain payment instruments with a payment location (via the PaymentOption entity). It also identifies the following:

- Whether the payment option is a preferred payment option for the service provider.
- What the payment option terms of payment are from receipt of the invoice to the expectation for payment (stated as a temporal duration), and if the service also provides a tax invoice subsequent to invocation.
- Whether there is a charge associated with a particular payment option.
- Whether a payment option is only available to requestors from a particular region.
- Whether there are conditions that surround the use of the payment option; and
- When the payment is due. We capture a payment schedule with respect to a payment option in Figure 4.15. This schedule can be represented using one of two approaches. Firstly, the stating of a percentage of the overall price, and the temporal relationship to the service provision act (i.e. before, during, or after). Alternatively, the percentage can be stated with a temporal entity that applies. This allows for one or more temporal entities, which caters for the description of multiple payments.

The following example query filters instances of payment options based on the following criteria: accepts credit cards and has terms of payment of 14 days.

PaymentOption
⊢ can be fulfilled using payment instrument CardBasedInstrument
⊢ has card type "Credit"
⊢ has invoice terms of payment of TemporalDuration
⊢ has cardinality 14
⊢ has StandardTemporalGranularity
⊢ has standard granularity name "Day"

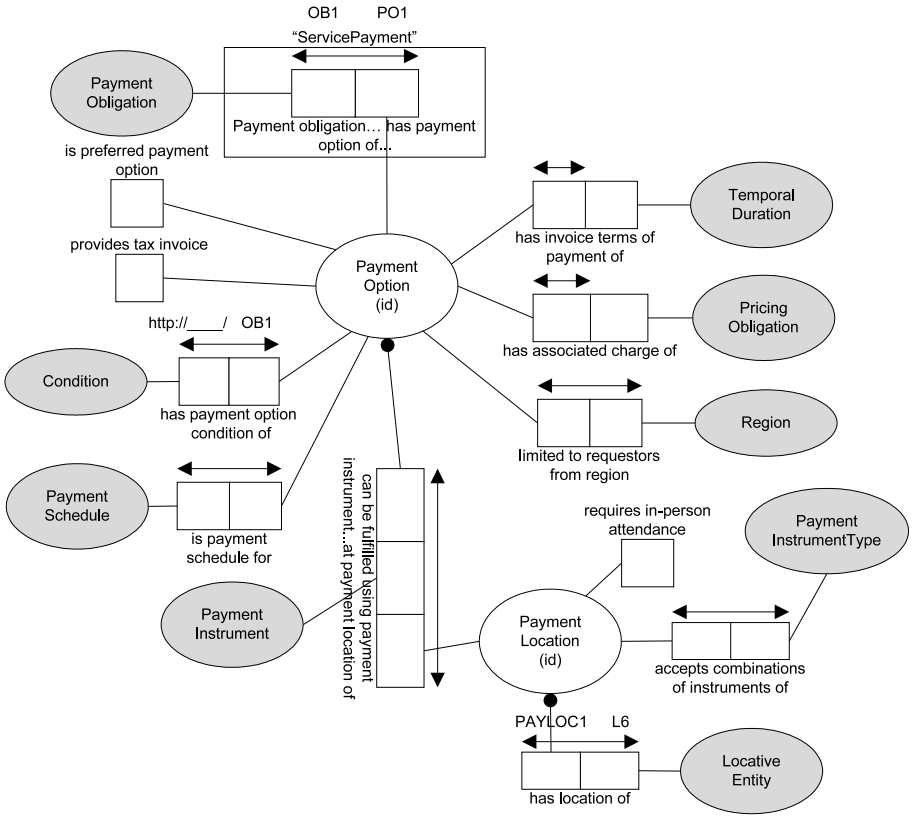


Figure 4.14: Payment options

The payment location captures where the payment is to be fulfilled. This concept utilises the notion of locative entities to allow payments to be directed to multiple locations, and to support different payment instrument combinations at each location. We can group payment locations using their type from the locative model. This allows us to provide easier conceptual querying of payment locations. Examples of payment location types include postal addresses, phone numbers and URIs. This allows us to answer conceptual queries such as - Find a service where the LocativeEntityType is "M" (referring to the PhoneNumber subtype of LocativeEntity) and the payment location is an Australian phone number (+61)? This type of query may be useful for people that have access to a telephone but who do not want to call an international phone number to provide payment.

The following example query filters instances of payment options based on the following criteria: accepts instruments at a location that is a region with the common name of "Melbourne", and that issues tax invoices.

PaymentOption
⊢ has payment location of PaymentLocation
⊢ has location of Region
⊢ has common name of "Melbourne"
⊢ provides tax invoice

The ability to capture payment conditions provides the option to attach a condition to each of the payment options. For example, some providers want you to spend a minimum of $10 for all credit card transactions. Others restrict you from paying bills over a certain value via certain forms of payment instrument.

4.3.1 Payment instruments

Our models for payment instruments are presented in Figures 4.16 and 4.17. We consider payment instruments to have some common properties that are outlined in the first model. Payment instruments are issued by a provider. This includes cash, which is an instrument normally issued by a Reserve bank within a country. Payment instruments support one or more currencies and have associated locative information. This includes the region it was issued in, and the region(s) the instrument's use is limited to.

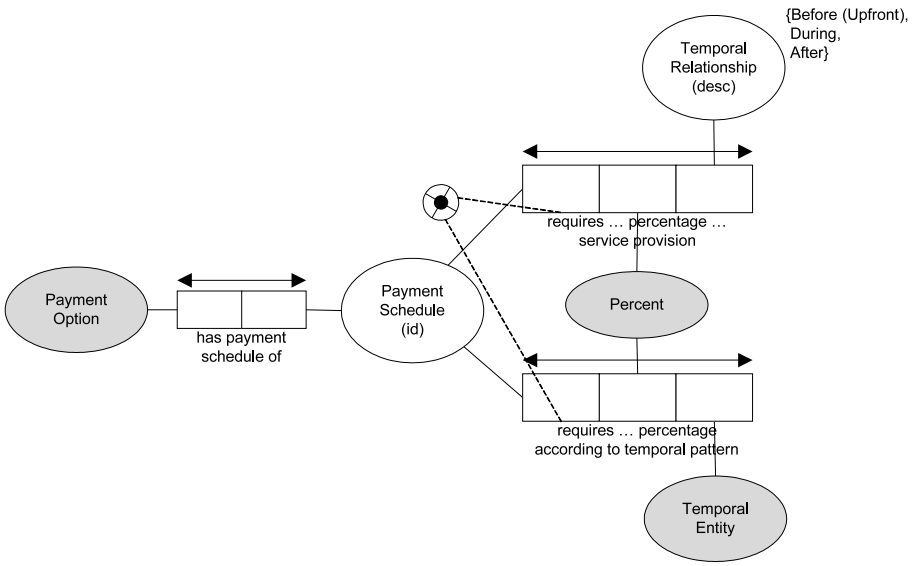


Figure 4.15: Payment schedule

Attached to a payment instrument we allow a surcharge to be stated. This can be used to support the notion of additional costs for certain instruments, for example credit cards, where the service provider charges an additional amount when the service requestor uses the credit card. We consider some payment instruments to support payment schemes, which are in turn controlled by a provider. For example, a payment instrument such as a credit card might support the "Visa" payment scheme.

We consider that payment instruments have four subtypes [see Figure 4.17]. These categories are defined as card based instruments, cheques, cash, and voucher based instruments. Cards include credit cards (e.g. Visa, MasterCard), charge cards (e.g. American Express and Diners), debit cards, store cards and stored value cards.

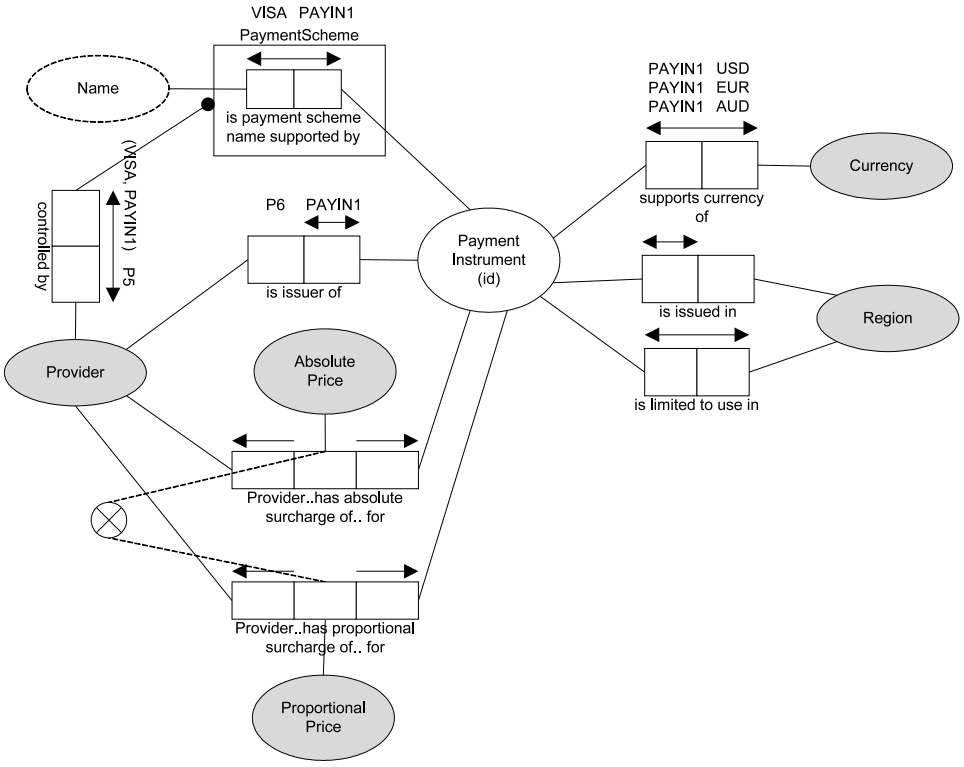


Figure 4.16: Payment instruments

We group cheques into one of four types - personal, bank, business and travellers. We consider cash to be further subtyped into electronic cash instruments. This refers to electronic transfers such as direct transfer, direct debit, digital cash and wire transfers. Vouchers should be stated as valid for redemption with a certain provider or service. They also have an associated temporal validity.

We expect that service requestors seek services based on the characteristics of payment instruments (e.g. traceability). A summary of payment instrument dimensions is available [60]. These dimensions (referred to here as characteristics) have been summarised to reflect the complex issues associated with their representation. We envisage that service catalogues will facilitate this discovery based on payment instrument characteristics as a form of value-add.

The payment instrument characteristics [see Figure 4.18] include:

- Offline: Can the payment instrument be used in a non-electronic environment?
- Online: Can the payment instrument be used in an electronic environment?
- Acceptability: The relative acceptance of the payment instrument by the receiving party (i.e. the service provider).
- Traceability: The service requestor, service provider (and any interim parties) and their associated operations/actions can be traced.

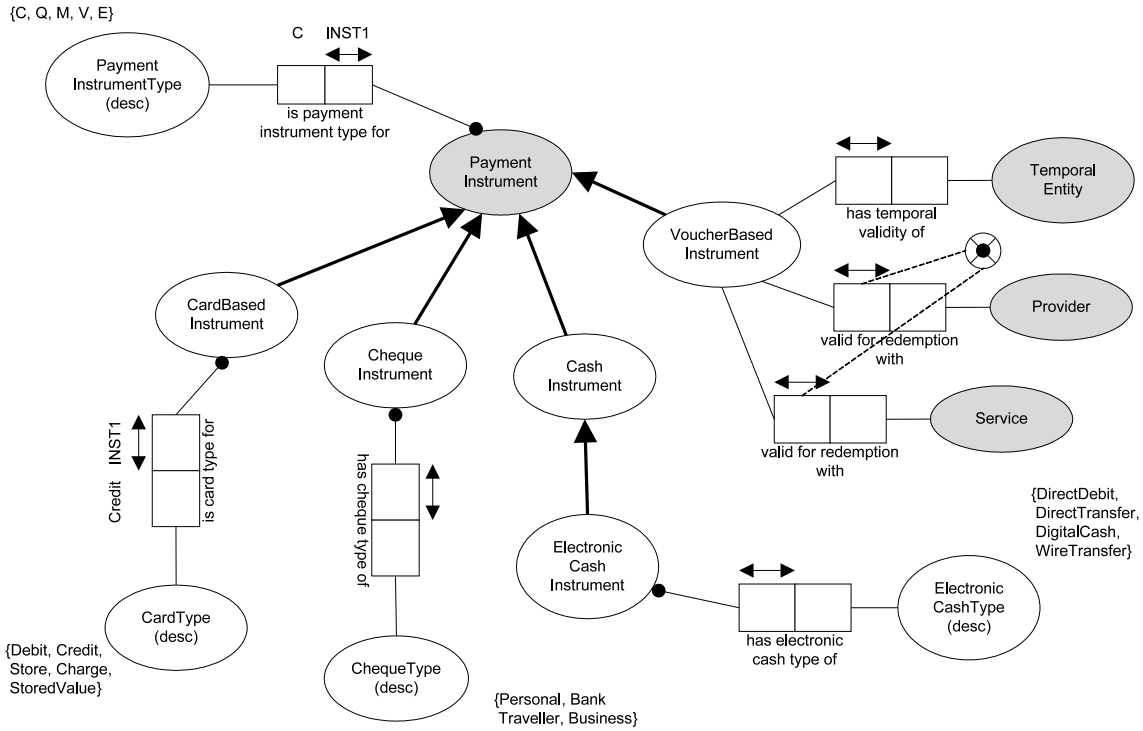


Figure 4.17: Payment instrument type hierarchy

- Refutability: Neither party is capable of denying either payment or service receipt.
- Negotiability: Does the payment instrument have the ability to alter the negotiating conditions associated with the commodity or service?
- Liquidity: Is it possible to liquidate a holding in the payment instrument within a short timeframe?
- Expiration: Does the payment instrument have a fixed lifetime?
- Provider Coupling: To utilise the payment instrument are you coupled (e.g. by way of a card, account, password loyalty program or PIN) to the provider?
- Transferability: Refers to the ease with which an instrument can be transferred to another instrument.
- Security: Does the use of the payment instrument occur in a secure environment?
- Immediacy: How quickly is the value of the payment instrument transferred from one party to another?

Table 4.2 outlines the relationship between payment instruments and payment characteristics. Understandably subjective in nature, it should also be noted that this table will be different depending on the context within which it is viewed (e.g. countries like the United Kingdom have a high acceptance of cheque payments. This may not be the case in non-cash centric economies such as Japan). We consider the populations within the table to be indicative of the values that catalogues could provide to assist with discovery based on payment instrument characteristics.

Payment Instruments	**Payment Characteristics**											
	Offline	**Online**	**Acceptability**	**Traceability** [2]	**Non-refutable**	**Negotiability**	**Liquidity**	**Expiration**	**Provider Coupling** [3]	**Transferability**	**Security**	**Immediacy**
Cash	◇ [1]		H	◇		◇	H			◇		I
Cheque	◇		H	◇		P [4]	M	◇	◇	◇		D
Direct Funds Transfer		◇	H	◇	◇	◇			◇	◇	◇	I
Credit/Charge Card	◇	◇	H	◇			◇		◇	◇		D
Traveller's Cheque	◇		M	◇	◇	◇	M			◇		D
Wire Transfer	◇		L	◇	◇		M					D
Money/Postal Order	◇		L	◇	◇		M					D
Security	◇	◇	H	◇	◇		M		P			D
Bond	◇	◇	L	◇			H			◇		D
Bank Bill	◇	◇	L	◇			H			◇		D
Voucher	◇		M	◇		P	H	P	◇			I
Stored Value Card	◇	◇	L	◇			L	P	◇			I
Digital Cash		◇	L	◇	◇		L		◇	◇	P	I
Anonymous Digital Cash		◇	L		◇		L		◇	◇	P	I

Table 4.2: Characteristics of payment instruments

[1] Legend : P = Possibly, H = High, M = Medium, L = Low, I = Immediate and D = Delayed. ◇ indicates the dimension is applicable to the payment mechanism.

[2] Traceability has varying degrees. Face-to-face cash transactions have a degree of tracability. However this type of transaction could also be conducted on behalf of someone else (e.g. by giving money to someone to buy something for you).

[3] Coupling includes items such as accounts, passwords, and personal identification numbers.

[4] Cheques can be marked "not negotiable".

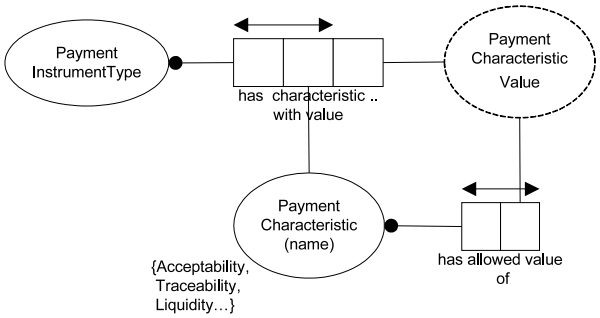


Figure 4.18: Payment instrument types

4.4 Reward schemes

Some service providers choose to reward service requestors using loyalty schemes. We view the redemption of rewards as a method for service payment in some circumstances [see Figure 4.19]. Reward schemes are an important way of maintaining customer loyalty and the redemption of such rewards is a sensitive issue. A number of reward points can be used during the redemption process. Like the accumulation of rewards, there are temporal constraints and associated conditions that control their use.

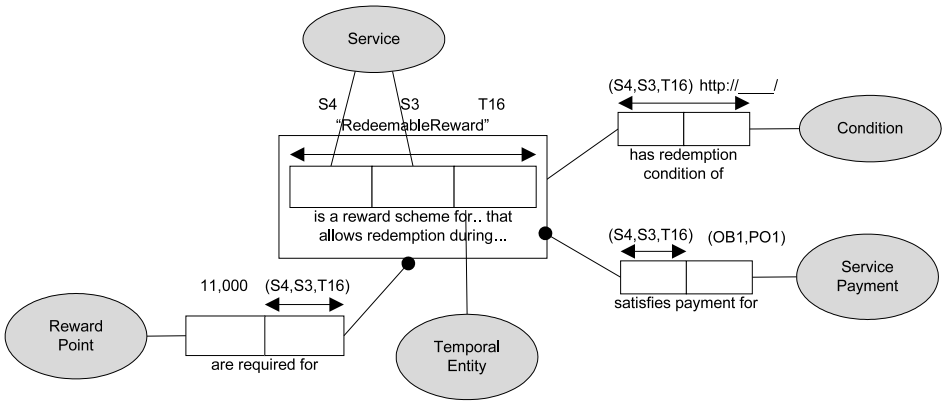


Figure 4.19: Service payment via rewards

We attach to the price of a service the possibility to accumulate rewards under a reward scheme [see Figure 4.20]. Reward schemes can be provided by the actual service provider or by a third-party service provider. Examples of reward schemes include frequent flyer programs, and credit card reward programs. Our model allows a service to attach a number of reward points to the invocation of the service, based on the service price that is paid (remembering that prices have a temporal and a locative availability). Reward points are only available during certain temporal intervals, or on a particular date, as well as having conditions attached.

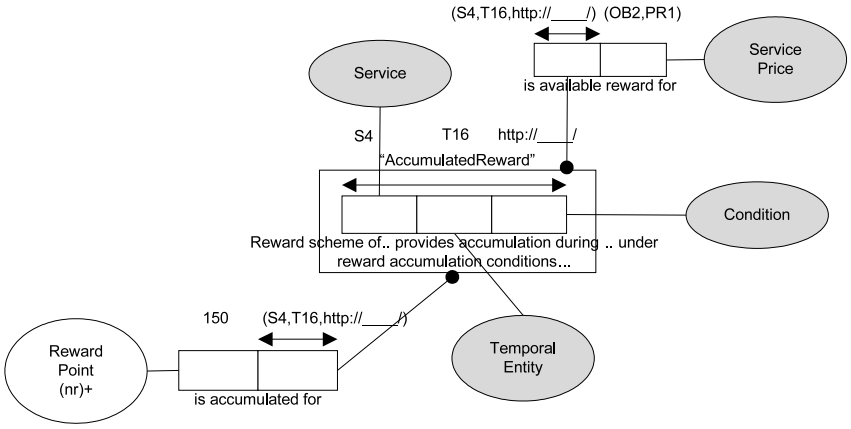


Figure 4.20: Service price specified as rewards

Summary

This chapter has presented a discussion of obligations, price and payment. These have been shown to be complex entities that are exceptionally rich with non-functional properties. We feel that these non-functional properties will be critical to enhancing the service discovery and comparison tasks. We believe that, whilst complex, we have managed to capture these entities within our conceptual models. In a related discussion we presented our conceptual models for reward schemes offering reason as to why they are linked to the concepts of payment and price.

The next chapter is a discussion of discounts, penalties and rights.

Chapter 5

Discounts, Penalties and Rights

This chapter is a discussion of properties that utilise price related entities to express some of their semantics. We use the term *discounts* to describe an approach that is regularly used within business in increase custom. We use the term *penalties* to describe the effect of non-compliance with a particular obligation of service provision. Finally, we refer to *rights* as the permissions granted to a service requestor or service provider as part of service provision.

5.1 Discounts

Discounting is a common method of attracting custom. Various types of discounts are available for services. We have previously shown in section 4.1 that pricing obligations and payment obligations may both attach their related discounts. We view discounts from the perspective of the service requestor, and therefore we believe that discounts can be categorised according to how you pay (e.g. early payment, coupon used), as well as to who you are (e.g. an elderly person) [see Figure 5.1]. We refer to this distinction as payment related discounts and payee related discounts respectively.

The service provider is unable to determine in advance all the combinations of service discounts that might apply to a price based on attributes of the service requestor (e.g. their age, membership to associations). For this reason, the catalogue provider (who may have more context related information with respect to the service requestor) may apply the discounts to a price before it is presented to the requestor during the discovery process. Our notion of discounts assumes that they are not included within the price specified by the provider.

A service provider might want to state the discount in one or two ways: as a reduction of the price of a service (e.g. 10% off the price), or as a resulting price for a service (e.g. $90 is the resulting price). In some cases it may not be the original service where the discount is available. A discount may be offered at an alternative service, perhaps also owned and operated by the same service provider. This allows a service provider to entice a service requestor with a discount on the price of a service, and to then provide a discount on another (possibly more expensive) service of the provider. We use the Price supertype (discussed in chapter 4) to

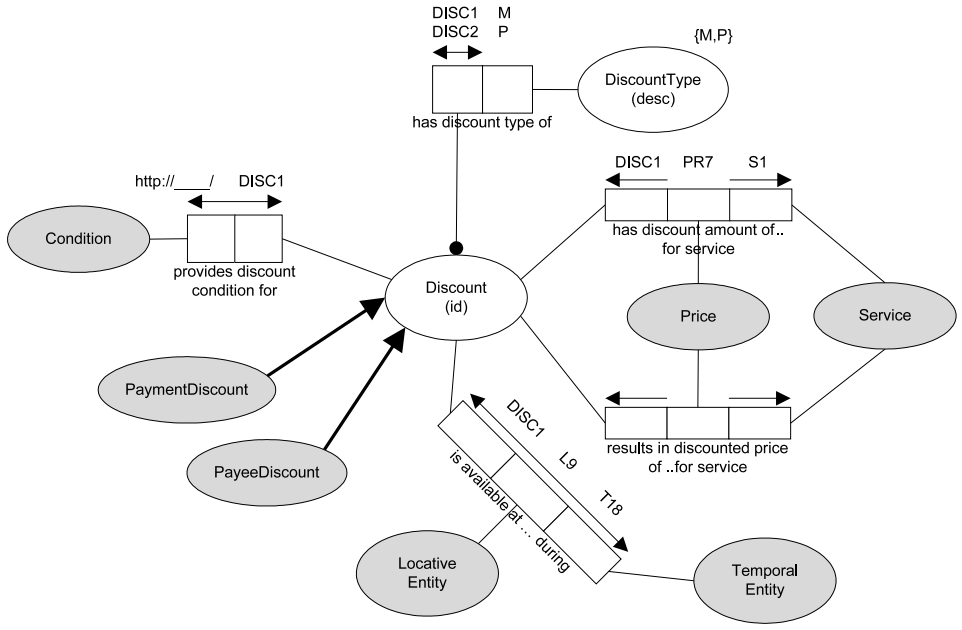


Figure 5.1: Discounts

allow for the inclusion of a range of possible discounts expressed using entities such as RangedAbsolute or RangedProportional price. For both payment and payee related discounts we allow the expression of the temporal and locative constraints. Service providers may also state any conditions associated with receiving a particular discount.

We consider that service providers who offer their services for a cheaper price during a specified period (i.e. a sale) are capable of expressing this fact using the temporal validity aspect of our price obligation model. The price validity may refer to a date range when the price is cheaper. For this reason we don't think of a sale as a type of discount. This approach to the modelling requires that a series of prices be specified; one for before the sale, one for during the sale and one for after. With the use of an appropriate user interface for entering service descriptions, the service provider could be largely spared from this problem. It should be noted that whilst not ideal, this approach does achieve the desired outcome.

Payment related discounts optionally require a minimum purchase amount before the discount can be received. The following are particular forms of payment related discounts [see Figure 5.2]:

- Early payment - this type of discount is offered to service requestors who are able to meet a payment obligation earlier than required. An early payment offset, that is a once-off period for receipt of the early payment, or an early payment schedule is captured.

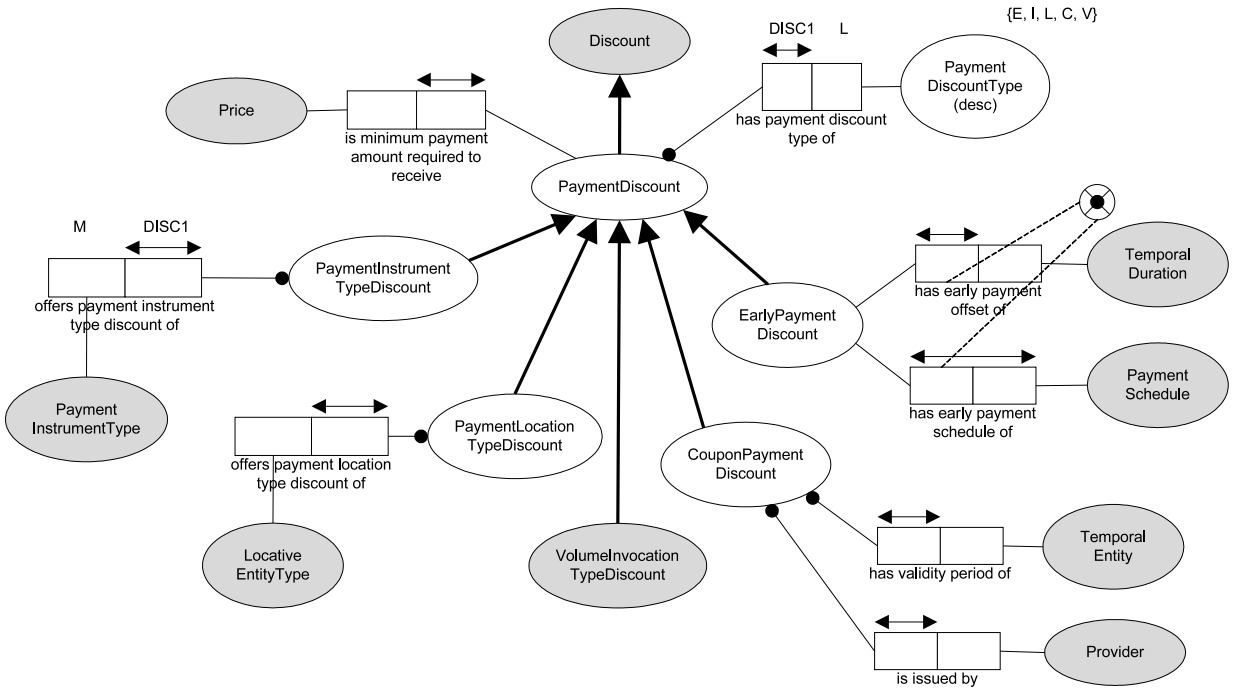
- Type of payment instrument - this discount is provided based on the type of payment instrument being accepted by the service provider. For example, a

service provider may offer a discount for payment using "Cash" as they are not incurring merchant fees associated with payment instruments like credit cards.

- Coupon/Offer - coupons are common paper-based mechanisms for advertising discounts for services. They are typically constrained to a specific temporal pattern (e.g. valid until a certain date, valid between certain dates). This type of payment discount does not include gift vouchers as we consider these to be a form of payment instrument.

- Payment location type - this discount is provided based on the type of payment location that is used by the service requestor. For example, a service provider may be willing to offer a significant discount for service requestors that utilise their Internet site. It may prefer to charge full price for the requestors who utilise a retail shop facility.

- Volume invocation - this discount is provided based on the number of invocations for a service that are submitted by a service requestor.



each EarlyPaymentDiscount **is a** PaymentDiscount **that** is of PaymentDiscountType 'E'
each CouponPaymentDiscount **is a** PaymentDiscount **that** is of PaymentDiscountType 'C'
each PaymentLocationTypeDiscount **is a** PaymentDiscount **that** is of PaymentDiscountType 'L'
each PaymentInstrumentTypeDiscount **is a** PaymentDiscount **that** is of PaymentDiscountType 'I'
each VolumeInvocationTypeDiscount **is a** PaymentDiscount **that** is of PaymentDiscountType 'V'

Figure 5.2: Payment discounts

The VolumeInvocationTypeDiscount is presented separately in Figure 5.3. Volume invocation discounts are intended to capture a reduced price for a service based

on the number of invocations of a service that are submitted by a service requestor. To provide for this type of discount we allow a service provider to state the number of invocations as a range. This ternary fact type is then objectified (in Figure 5.3) into an entity called VolumeInvocationRange. To this objectified entity we then attach a price, and optionally, a period of validity. This period of validity is expressed as a temporal entity (e.g. a duration, an anchored temporal instant).

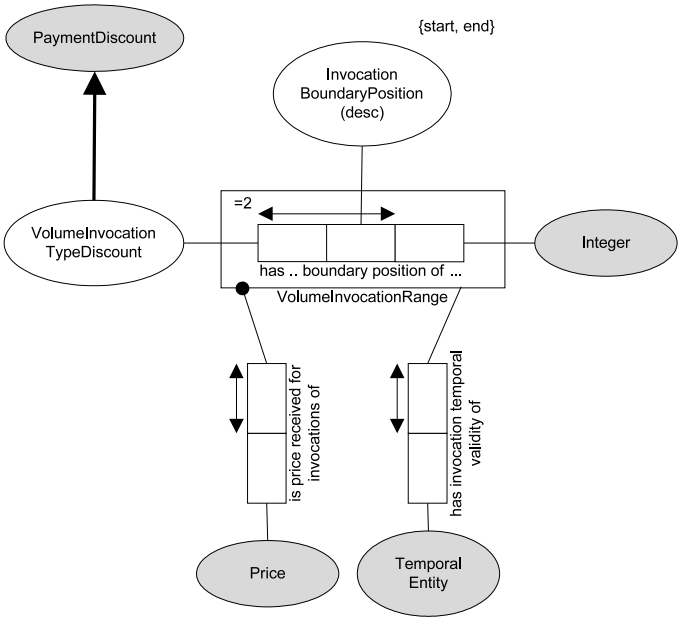


Figure 5.3: Payment discounts - Volume invocation

The following example query filters instances of payment discounts based on the following criteria: the price discount is an absolute price type, is a maximum of $50 Australian, and is for early payment at least 14 days in advance.

EarlyPaymentDiscount
- ⊢ has discount amount of AbsolutePrice
  - ⊢ has amount <= 50.00
  - ⊢ has currency “AUD”
- ⊢ has DiscountType “PaymentDiscount”
- ⊢ has early payment offset of TemporalDuration
  - ⊢ has cardinality >= 14
  - ⊢ has StandardTemporalGranularity of
    - ⊢ has standard granularity name “Day”

The second form of discount, payee discounts, relates to who the service requestor is. The following are payee related [see Figure 5.4]:

- Age group - service requestors who belong to a specific age group often find that they receive a discount. This is common for pensioners/seniors, infants, and children.

- Student - discounts are sometimes offered based on the requestor being a student. We provide differentiate between school and full-time university students for more expressiveness.
- Membership of a particular service sometimes brings with it discounts with another service provider. Large organisations (e.g. health funds) whilst requiring membership themselves, normally negotiate discounts with other service providers on behalf of their members.
- Shareholder - discounts are sometimes provided to shareholders. There is normally a requirement for holding a minimum number of units before the discount is available.

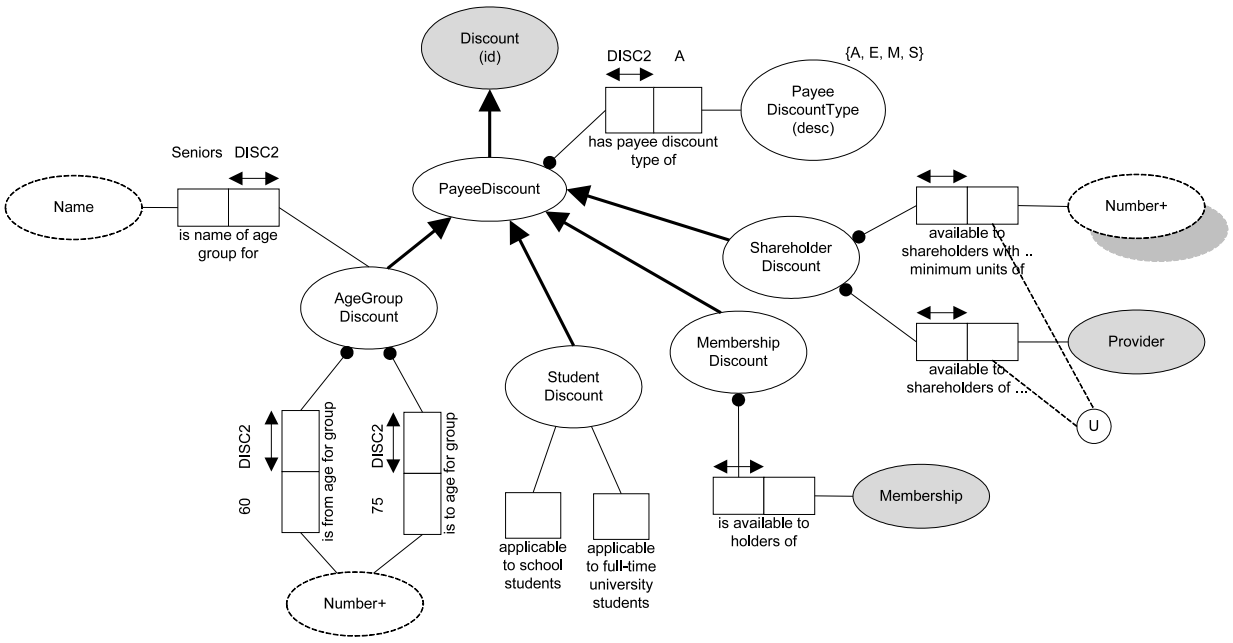
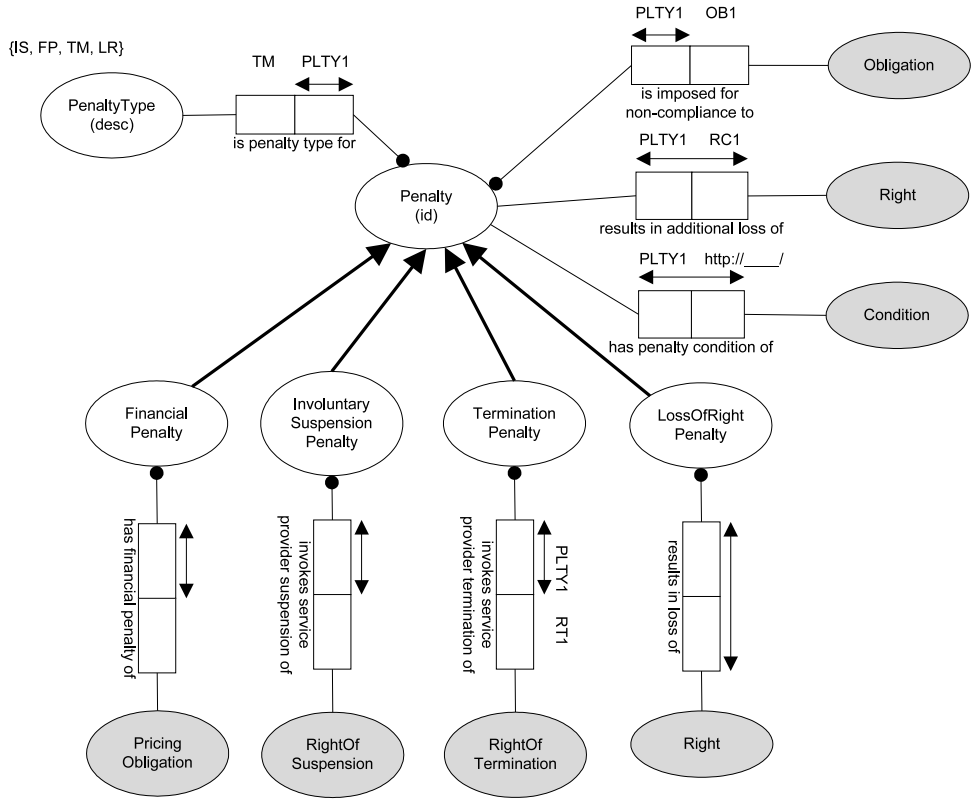


Figure 5.4: Payee discounts

5.2 Penalties

A penalty is a mechanism for service providers to describe what will occur in the event that a service requestor does not comply with a specific obligation. Penalties are commonly outlined in service level agreements as a means of compensating the service requestor for non-performance (in the generic use of the term performance). An example of a condition under which a penalty is applied is for non-payment or late payment by the service requestor. Penalties will normally have a related set of conditions.

Figure 5.5 presents our model for penalties. We provide specific subtypes for four types of penalties: termination, financial, involuntary suspension and loss of right. By termination we refer to the service provider ceasing to provide to the service requestor the output of the service. Termination is non-reversible. Our model for penalty introduces a link between the TerminationPenalty entity and a particular form of right (rights are discussed in further depth in section 5.3) that we refer to as the RightOfTermination. This right of termination ensures that the provider does not have to honour its previous agreement with the service requestor.



each InvoluntarySuspensionPenalty **is a** Penalty **that** is of PenaltyType 'IS'
each FinancialPenalty **is a** Penalty **that** is of PenaltyType 'FP'
each TerminationPenalty **is a** Penalty **that** is of PenaltyType 'TM'
each LossOfRightPenalty **is a** Penalty **that** is of PenaltyType 'LR'

Figure 5.5: Penalties

We consider a financial penalty to be straightforward. Under this scenario, the service provider chooses to impose a financial punishment on the service requestor. This is represented in our model as a PricingObligation that is defined by the service provider. Involuntary suspension refers to the service provider's decision to temporarily interrupt the provision of a service to a particular requestor. Involuntary suspension within our model is represented as a link to the RightOfSuspension entity that is outlined in section 5.3.3. For the moment it is sufficient to say that involuntary suspension is for a specified period, and has conditions, procedures and/or obligations surrounding the suspension and resumption.

Finally, penalties may result in the loss of one or more rights. This type of penalty can be used to capture the loss of warranty rights, the loss of access to a service, the loss of the right to recourse etc. Although the involuntary suspension and termination penalties are linked to their associated rights, we have chosen to represent them as distinct subtypes of penalty to achieve greater discovery capability.

The following example query filters instances of penalties based on the following criteria: the penalty is imposed for not complying with a payment obligation, is an involuntary suspension that has a maximum suspension period of 10 days, and has an obligation on resuming that is a payment obligation for not more than 50 Euros.

InvoluntarySuspensionPenalty
- ⊢ is imposed for non-compliance to PaymentObligation
- ⊢ invokes service provider suspension of RightOfSuspension
  - ⊢ has maximum suspension period of TemporalDuration
    - ⊢ has cardinality <= 10
    - ⊢ has StandardTemporalGranularity of
      - ⊢ has standard granularity name “Day”
  - ⊢ has resumption obligation of PaymentObligation
    - ⊢ has base charge of PricingObligation
      - ⊢ has price of AbsolutePrice
        - ⊢ has amount <= 50.00
        - ⊢ has currency “EUR”

5.3 Rights

Rights are permissions granted to either the service provider or the service requestor in the environment within which the service operates. We have previously shown that rights can be used in two contexts. Firstly, that along with a service price there may be associated rights available to the requestor or the provider. Secondly, that the penalties for not meeting the obligations associated with a service can result in the loss of one or more rights of the service.

Rights include the following: access to service resources, recourse (or appeal), suspension/resumption, termination, privacy, warranty (or guarantee), refusal of service, disclosure, cooling off periods, liability limitation and extension to the original service provision commitment [see Figure 5.6]. We attach a name, a period of validity and a designator that outlines whether the rights are for the service provider or the service requestor.

5.3.1 Access

Access refers to the right to the use of the service, normally for a specific period [see Figure 5.7]. This temporal validity is inherited from the “Right” supertype presented in Figure 5.6. The right is modified to include an access type. Access

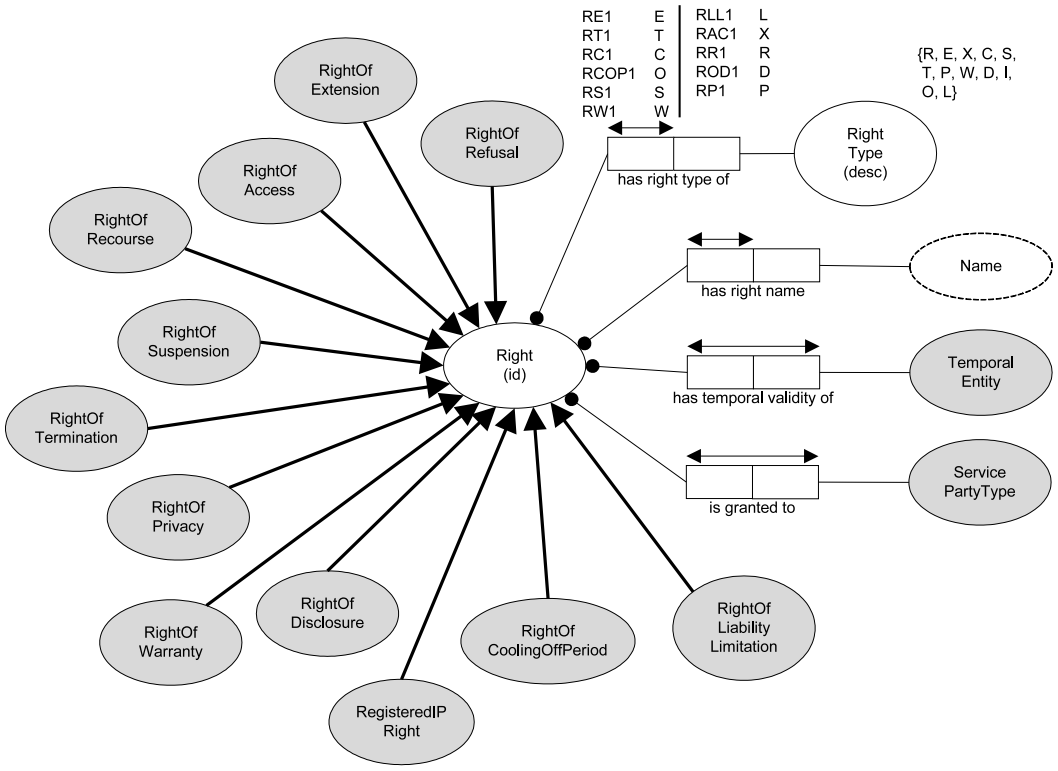


Figure 5.6: Rights

types describe a continuum of permission to use a resource and include exclusive, restricted, shared, and prohibited access. Exclusive access is normally only available when the service requestor incurs an obligation (e.g. a higher than normal payment, or an extended relationship commitment between both parties) that is additional to the standard obligations for a service offering. We consider this right of access to be distinct from security. Security, in this thesis, relates to the physical approaches that are taken to ensure the integrity, confidentiality, authentication and non-repudiation of callers to the service.

Access is provided to the service requestor over a resource that we identify according to a locative entity. This locative entity will typically be a URI. Optionally, we also identify resources according to a name and a type. Resource types include intellectual property, information/knowledge, a design (e.g. a registered trademark), person(s), a facility, and time. These are intended to be indicative of the enumeration constraints for the ResourceType entity. The access to these resources is normally provided for a specific temporal duration that incurs one or more obligations.

5.3.2 Recourse

Requestors of a service have the right to recourse, or an appeals process [see Figure 5.8]. Recourse is usually available for a certain period of time after the service provision has been completed. Within this period, the service requestor can utilise

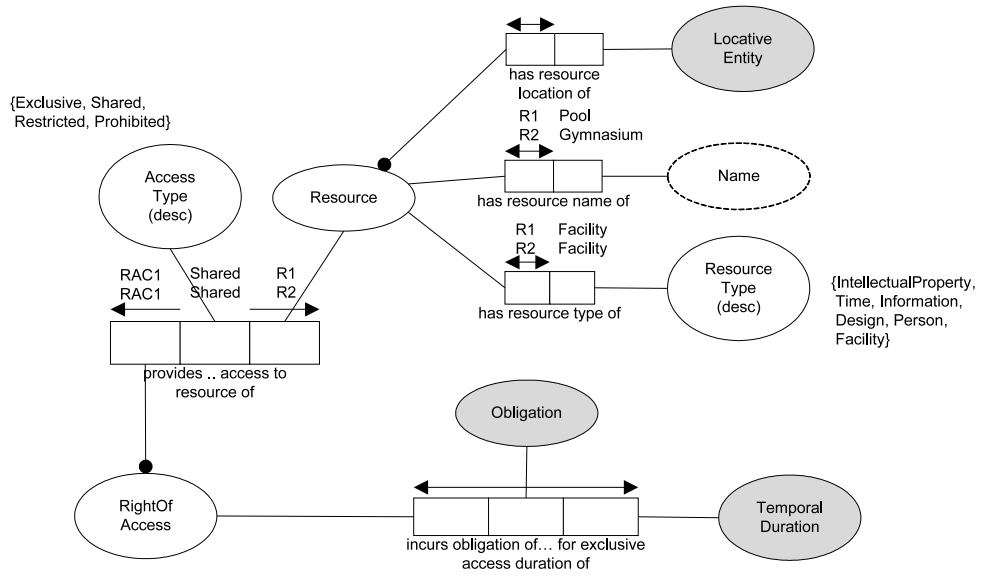


Figure 5.7: Right to access

an appeals procedure in an attempt to rectify the matter at issue. Recourse is sometimes mediated through a separate third party provider. In the case where it is not mediated it is assumed that the service provider controls the process of recourse.

In a service environment, recourse is administered according to the laws of a jurisdiction. The jurisdiction refers to a governed locative entity (e.g. a state within a country) and legislation within that governed locative entity. Each piece of legislation is referred to according to an assigned name and a year of introduction. We also capture that legislation is superseded or amended by one or more pieces of legislation.

5.3.3 Suspension and resumption

We consider that two types of suspension exist, voluntary and involuntary. Voluntary suspension is the ability of the service requestor to temporarily halt the provision of a service [see Figure 5.9]. Involuntary suspension results from the failure of a service requestor to meet obligations associated with a service. Involuntary suspension was outlined in section 5.2 relating to penalties. Both types of suspension may have an associated set of conditions, possibly incur a suspension or resumption charge, and have a procedure that outlines either or both the suspension or resumption process.

Within this section we are particularly interested in voluntary suspension. Voluntary suspension may be defined for specific periods of time. A service provider may set a minimum suspension period, a maximum suspension period, a maximum number of suspensions or even a maximum aggregated period of suspension that are available during the service provision process. The suspension of a service may

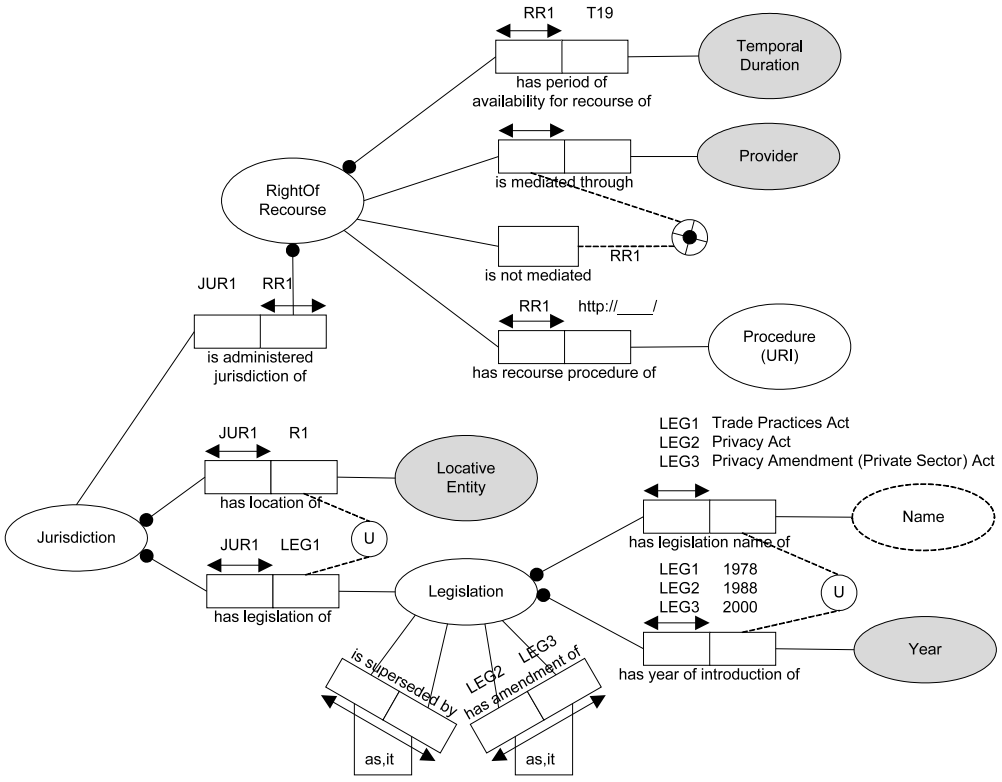


Figure 5.8: Right to recourse

normally be initiated at a certain locative entity, within the bounds of a temporal entity. Involuntary suspension results from the failure to meet one or more obligations. This type of suspension is invoked by the service provider under a set of conditions. The resulting suspension is normally for a specific suspension period.

5.3.4 Termination

Like suspension of a service, termination of the provision of a service can be initiated by either the service provider or the service requestor [see Figure 5.10]. Service providers will normally attempt to terminate a service commitment to a requestor when the requestor fails to comply with the obligations for a service. As with suspension we are particularly interested in termination from the voluntary perspective. Termination for failure to comply with an obligation is outlined in section 5.2 in relation to penalties.

Termination may incur an obligation (e.g. a cost), and may have a set of conditions surrounding its execution. The execution of the termination is normally governed by a termination procedure. The commitment to a service sometimes requires that termination be pre-conditioned by notification of the termination. This ensures that the party requesting the termination provides notification of the termination according to some temporal duration (e.g. 30 days notice).

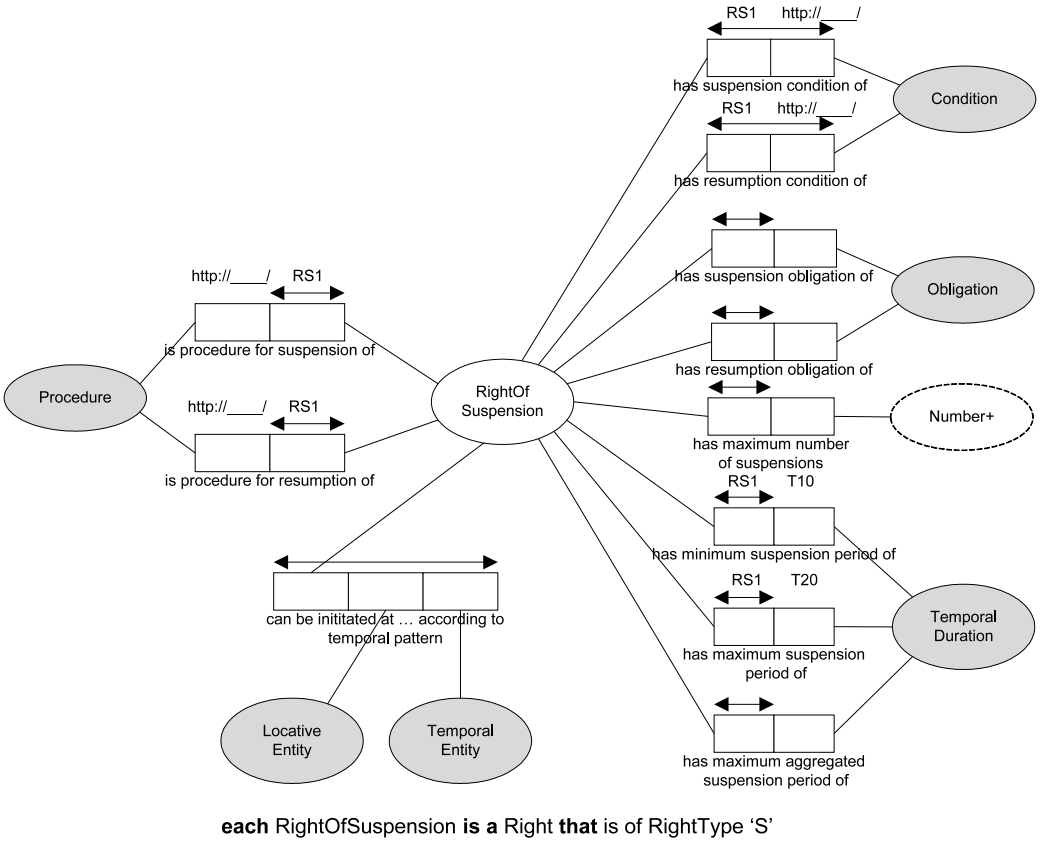


Figure 5.9: Right to suspend

5.3.5 Privacy

Privacy is primarily the concern of the service requestor who wants to ensure that the service provider appropriately deals with information disclosed by the requestor to the provider [see Figure 5.11]. Privacy legislation sometimes binds the actions of a service provider when dealing with service requestor related information. These privacy concerns may be captured in legislation (e.g. a privacy act) or through a corporate privacy statement of the service provider.

Privacy policies may be specifically defined for four distinct phases in the handling of service requestor information. These phases are the collection, storage, access and alteration of service requestor information. Concerns relating to a service provider's privacy commitment can normally be addressed at a specific location, within the bounds of a temporal entity. Service providers can outline other service providers who will be provided with certain requestor related information as part of the service provisioning. These secondary service providers may be subsidiary companies of the original service provider. Privacy of disclosed items relating to the service requestor may be for a specific period. Subsequent to the expiry of that period the requestor must be consulted about the disclosure of the information.

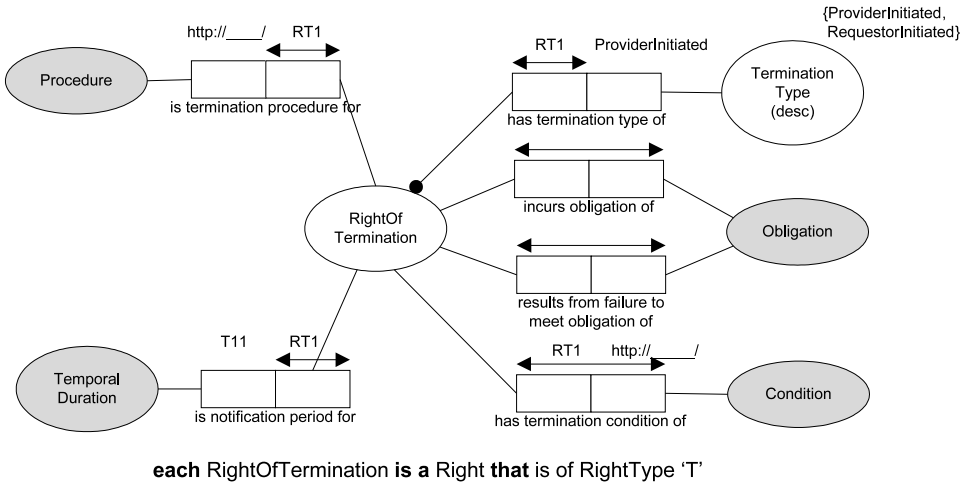


Figure 5.10: Right to terminate

5.3.6 Refusal of service

The right of refusal of provision is available to service providers [see Figure 5.12]. Service providers may outline the conditions under which service provision is refused, or may consider their right of refusal discretionary. The refusal of service provision may be enacted under a refusal procedure. Some service providers may offer an appeals procedure for requestors who are refused service. This procedure is normally available at a certain locative entity, within the bounds of a temporal entity.

Refusal of provision of a service is not to be confused with security of the service. When we refer to refusal we refer to notions such as nightclubs that retain the right to refuse entry to their services if patrons are inappropriately dressed, or who may have consumed excessive amounts of alcohol prior to their attempted entry.

5.3.7 Disclosure

Service providers occasionally request disclosure of information relating to the service requestor [see Figure 5.13]. This may be to allow them to create a database of target clients, or it may be to reduce their risk or exposure via a single client (as is the case with insurance providers). We refer to the information of interest to the service provider about the service requestor as disclosure items.

Some service providers may expect to be allowed to disclose requestor income, credit history, employment status etc. These facts about a service requestor are considered a disclosure item. The same entity "DisclosureItem" is used in a privacy context to outline the information about a requestor that is subject to privacy regulations/commitments. In this context it is primarily the service provider making it clear that to use the service, requestors need to provide some information about themselves. We define this DisclosureItem entity using a URI.

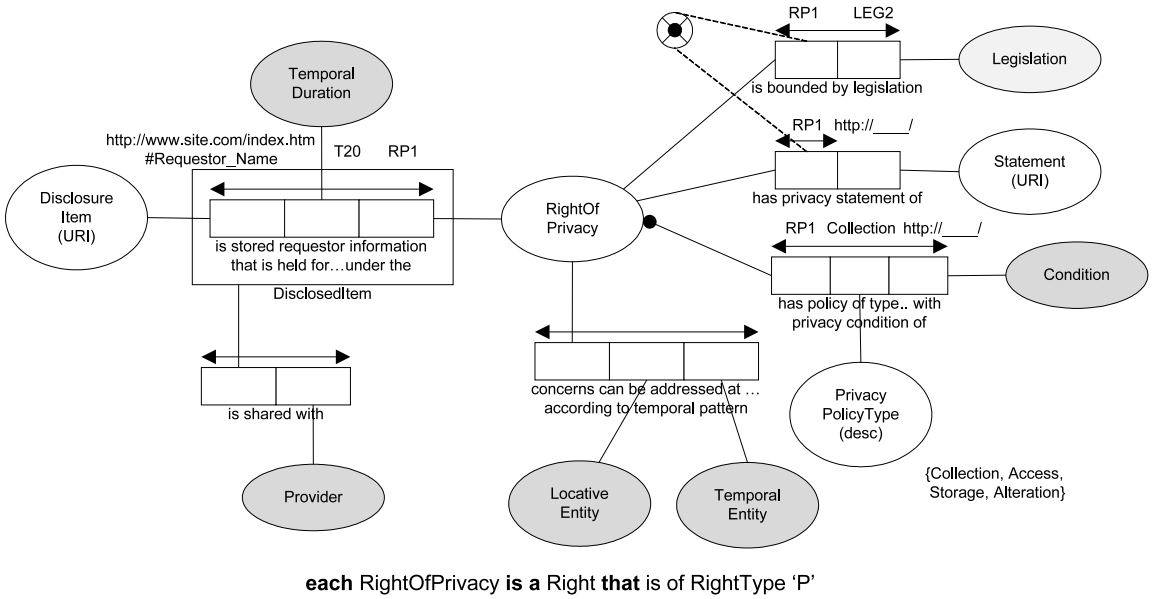


Figure 5.11: Right to privacy

5.3.8 Extension

The right of extension allows the service provider to advertise that provision may be extended for a certain duration under specific conditions [see Figure 5.14]. Extension to the provision may incur additional obligations (e.g. a lengthening of the original relationship commitment, or a cost), and is normally initiated via a defined procedure.

Extension is common with leasing arrangements (e.g. for office space), and provides the service requestor with the option to exit their agreement should they not be satisfied.

5.3.9 Warranty

Warranties (or guarantees) enable a service provider to reduce the uncertainty (that a service requestor has) surrounding the quality of service provision [see Figure 5.15]. Warranties are provided for specific item(s) and a specific period after the completion of service provision. They are surrounded by conditions, and may be fulfilled by a provider other than the original service provider.

Warranties are normally initiated according to a procedure that is invoked at a specific location (i.e. a locative entity), and within some temporal parameters (e.g. a defined temporal interval). Warranties may be revoked when the obligations that a service requestor has with a particular service provider are not met (e.g. payment).

5.3.10 Cooling off period

Service requestors want a level of protection after requesting a service. This is referred to as a cooling off period [see Figure 5.16]. Normally, if a service offers a

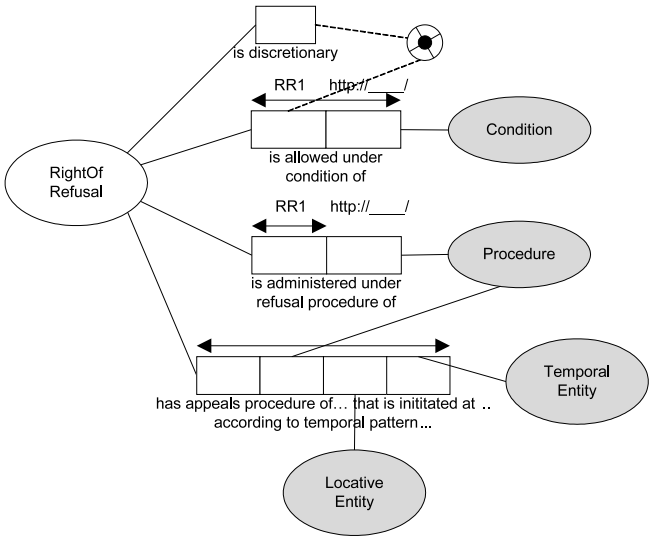


Figure 5.12: Right to refusal of service

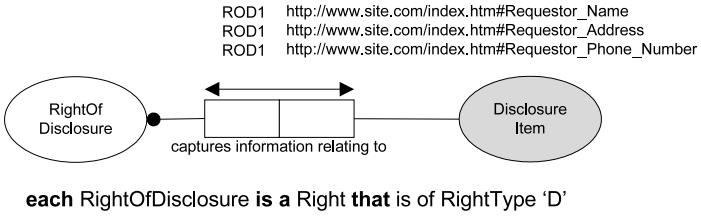


Figure 5.13: Right to disclosure

cooling off period the service requestor will be able to annul the service provision that they previously requested. By the term "annul", we are referring to the cancellation of all obligations that requestors have with the service provider. This annulment is normally available for a temporal duration (e.g. 7 days) after the request is made, or into the provision of the service. The latter is useful for services where the request and provision are non-contiguous. Our model also captures the conditions surrounding the cooling off period, and the procedure that needs to be invoked to ensure annulment of the service provision occurs.

5.3.11 Liability limitation

Service providers want a level of protection from liability in the event of failure to provide a service as promised to the service requestor. We refer to this as limitation of liability [see Figure 5.17]. Our model provides an ability to capture the conditions under which the limits to liability can be expressed to the service requestor.

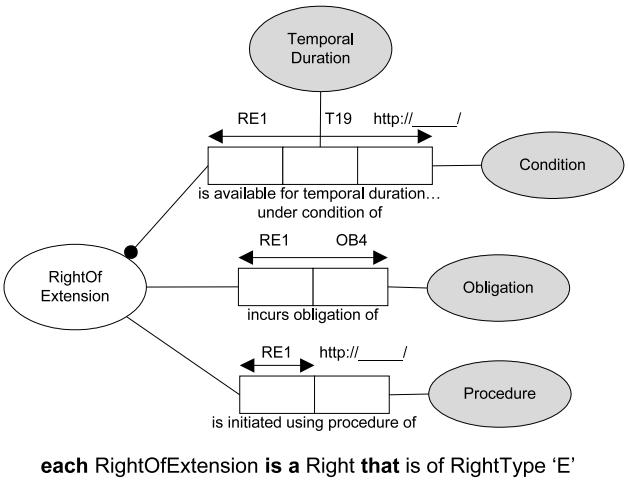


Figure 5.14: Right to extension of service provision

5.3.12 Registered intellectual property

Rights may take the form of registered intellectual property (IP) rights that a service provider holds. This includes common intellectual property rights such as patents, trademarks and designs. We discuss registered intellectual property rights generically before dealing with the specific subtypes of trademarks, patents and designs. All registered property rights are owned by a service provider. They will commonly have the following related details [see Figure 5.18]:

- An application number (assigned when it is lodged).
- A status for the application of the intellectual property right. Applications for intellectual property rights move through various statuses. These indicatively include applied, published, examined, approved and rejected.
- A lodgement date on which the application for registration of the intellectual property right was lodged.
- A registration date on which the application was granted for the intellectual property right.
- A link to a more detailed description of the intellectual property right. Normally this would be a URI pointing to a patent/trademark agency such as the United States Patent and Trademark Office.
- The country within which the intellectual property right originated.
- The area (i.e. region) within which a granted intellectual property right applies.
- An agent (normally legal) for the party making the application.
- The address of the agent for the party making the application.

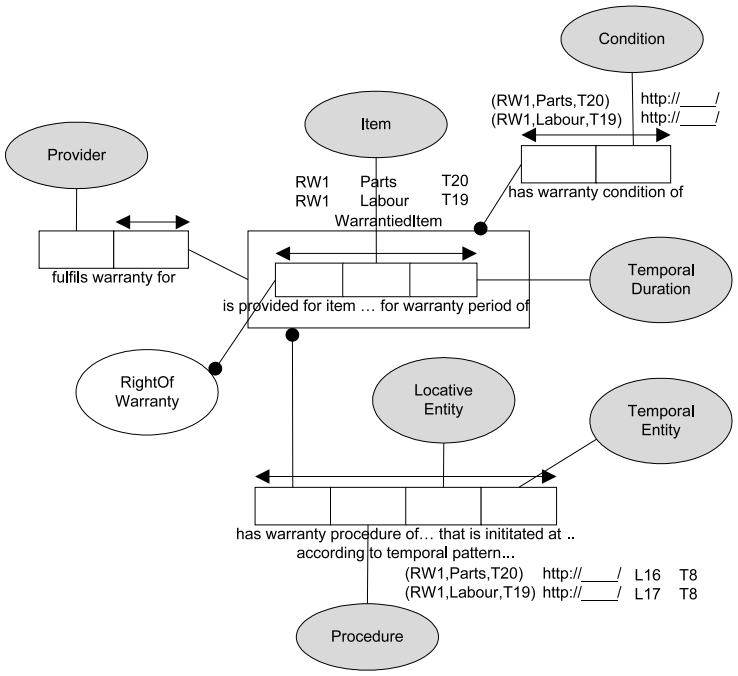


Figure 5.15: Right to warranty

Instances of a registered intellectual property right are one of the following types: patent, trademark or design [see Figure 5.19]. We also include a TrademarkOrDesign entity that captures the duration of protection granted to trademarks and designs. This property is inherited by both trademarks and designs.

Patents are official rights granted to an inventor with respect to an invention [109]. Within our patent model we capture the name of the inventor of the patent, and the title of the patent. We provide support for the classification of patents according to the Strasbourg Agreement Concerning International Patent Classification [118]. This agreement, signed by member countries, provides for a classification of patents according to a hierarchical identifier that includes a section (identified using letters A - H), a class (identified using a two digit number), a sub-

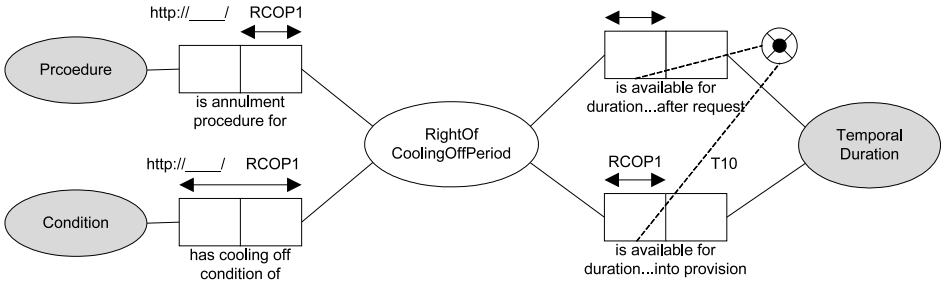
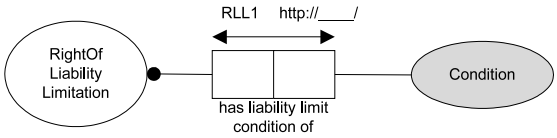
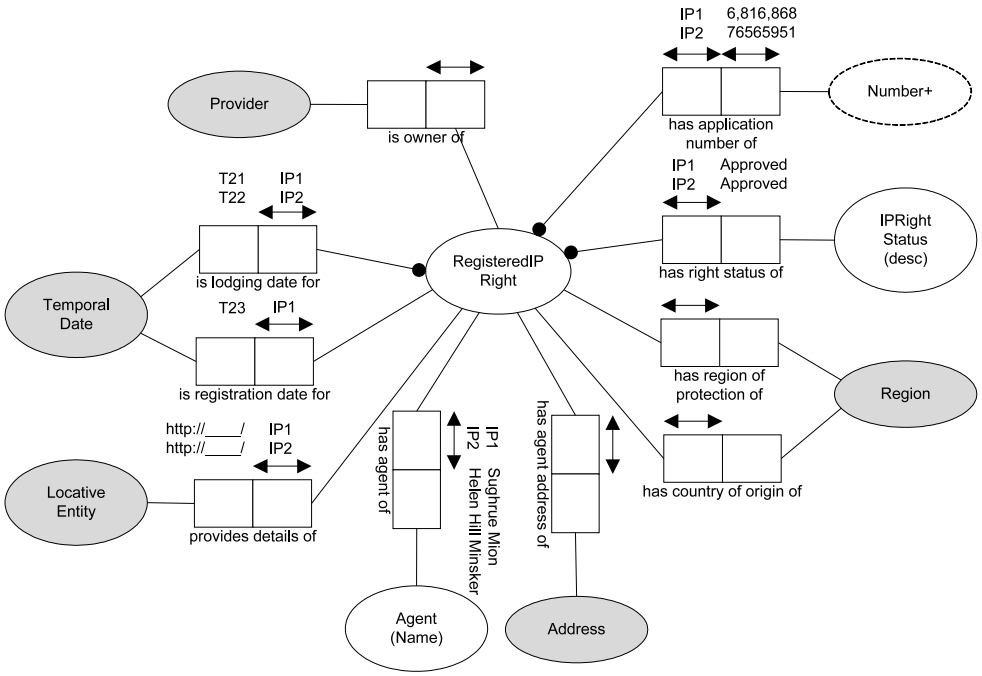


Figure 5.16: Right to cooling off period



each RightOfLiabilityLimitation **is a** Right **that** is of RightType 'L'

Figure 5.17: Right to limit liability



each RegisteredIPRight **is a** Right **that** is of RightType 'I'

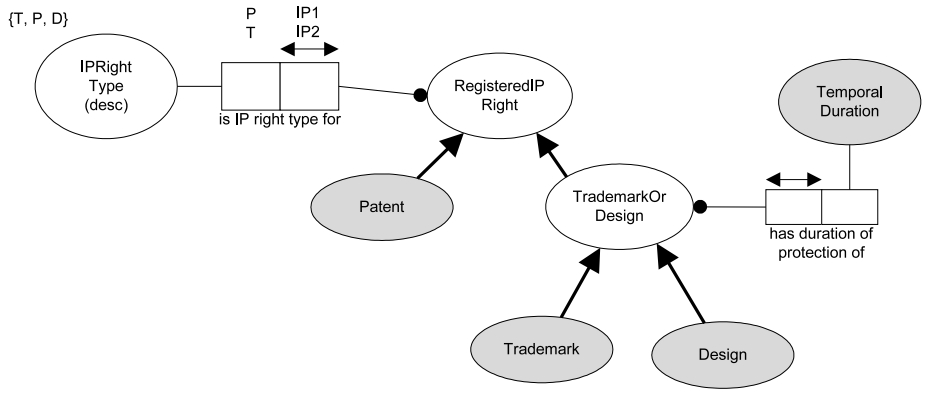
Figure 5.18: Registered IP rights

class (identified by a capital letter), and a group (identified using two, two digit numbers separated with a forward slash).

A trademark is "a word, name, symbol or device that is used in trade with goods to indicate the source of the goods and to distinguish them from the goods of others" [109]. Trademarks are further divided into the categories of trademark, servicemark, dressmark, collectivemark or certification mark. Servicemarks are similar to trademarks except that they apply to a service, not a product. Trademarks have a wordmark that is outlined at a URI location.

Classification of trademarks is supported through the figurative marks contained within them [117], and the goods or services that the trademark applies to [116]. The former classification scheme uses a category number (1 - 29) and division (identified by a two digit number). The latter approach involves 45 categories, the first 34 of which are related to goods. The remaining categories relate to services.

A design consists of the "overall appearance of a product", that when considered with respect to its "shape, configuration, pattern, and ornamentation" give it a "unique appearance" [51]. Design may also be classified according to the In-



each Patent **is a** RegisteredIPRight **that** is of IPRightType 'P'
each Design **is a** TrademarkOrDesign **that** is of IPRightType 'D'
each Trademark **is a** TrademarkOrDesign **that** is of IPRightType 'T'

Figure 5.19: Registered IP right subtypes

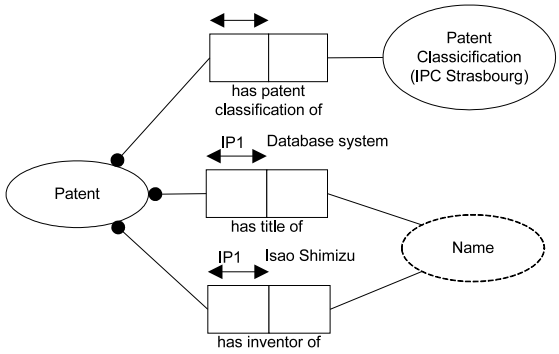


Figure 5.20: Patents

ternational Design Classification [119]. The scheme, referred to as the Locarno classification, provides a 32 class and 223 sub-class categorisation.

Summary

This chapter has presented a discussion of discounts, penalties and rights. These non-functional properties are complementary to those of payment and price presented in the previous chapter. We have a particularly rich model for capturing rights that are available to both service requestors and service providers. This richness can be explored in considerable detail during service discovery, without the need to contact the service provider. We believe this to be advantageous to the discovery process, as it shortens the time that the requestor spends undertaking discovery.

The next chapter is a discussion of security, trust and quality. These properties have been grouped together due to their ability to impact the perception of service requestors.

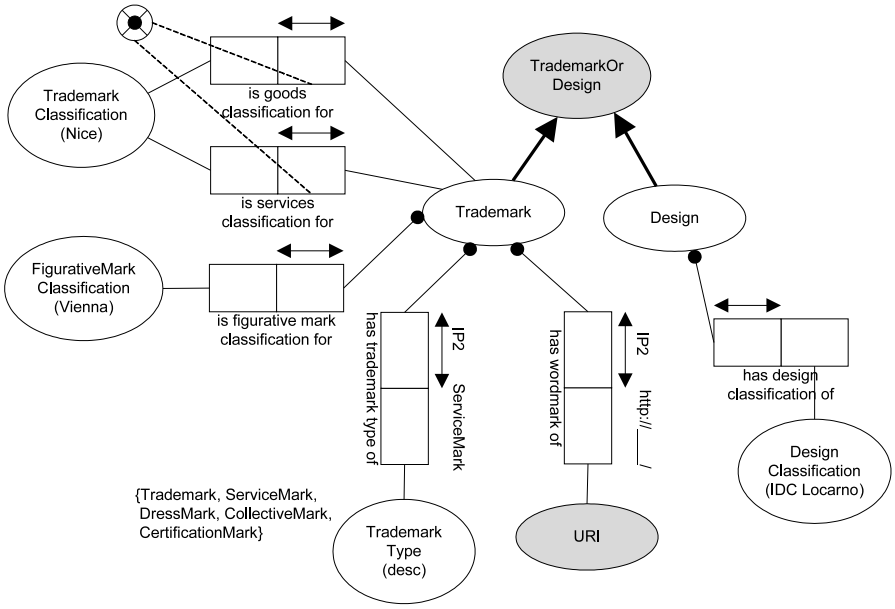


Figure 5.21: Trademarks and designs

Chapter 6

Trust, Quality and Security

In this chapter we deal with non-functional properties that have a high level of effect on our perception of both the service provider and the service. Trust, quality and security contribute to the feelings of well-being that a service requestor has with respect to a service. Traditionally, these properties have not been included in the description of a service. We argue that their inclusion is an essential part of any description of a service. We also believe that their expression within a service description will dramatically increase the level of comfort that a service requestor prior to invoking a service. We capture trust, quality and security in a number of models, but also indirectly through other models within our research. This is explained in more detail in the sections that follow.

6.1 Trust

Our view is that trust is largely indirectly represented in the models that are outlined in this paper. We believe that people's understanding and requirements for trust vary. Accordingly, some of the items that we feel contribute to a person's level of trust when dealing with both the service provider (in general) and the service include:

- Endorsement received from previous service requestors, or other service providers who compose services using the service in question.
- The cost of the service in comparison to the charge being levied by another service provider.
- The security of the payment instrument to be used (e.g. some credit cards offer a chargeback facility).
- The security of the location for payment.
- The support that a service has for certain quality standards.
- The membership of the service provider to a certain body (perhaps a professional body).

- The number of years that the service provider has been in business, or the number of years that the service has been offered.
- The mission statement of the service provider.
- The laws that govern the business operating environment.
- The links that this provider has with other providers who are perhaps better known to service requestors.
- The privacy of the information supplied to the service provider.
- The level of detail that the service provider makes available during the description and publishing process.

We choose to directly represent the notion of endorsement as a means of providing "referrals" from service requestors (or perhaps service providers) to the service in question [see Figure 6.1]. We consider endorsements to include the capture of the service party (presented in Figure 2.1) providing the endorsement, some comments relating to the service party's use of the service, a temporal instant when the endorsement was captured (providing context), and a subjective rating of the service provider/service by the party giving the endorsement. Ratings are discussed in more depth in section 6.2.1.

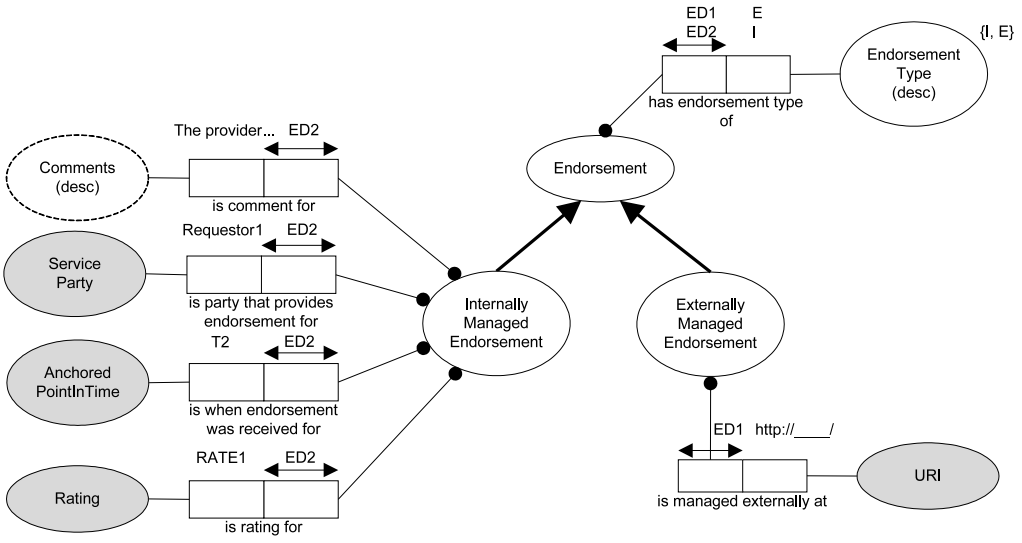


Figure 6.1: Endorsement

We view endorsements as being either managed internally by the provider offering the service, or being managed externally. The external management may imply a less biased view of the endorsements for a service. A provider who internally manages endorsements may be inclined to only show those that discuss their service

in a favourable manner. For those services where their endorsements are managed externally, we provide a link to the location where those endorsements are managed. It is likely that this would be used to refer to a URI.

We directly represent endorsements in our model by linking it to both the service provider and the service. As well as supporting the notion of endorsements for service providers [see Figure 6.2], we also capture the legislation that a provider is legally bound to comply with, the year that the service provider began business, the mission statement of the provider, the type of associations that it has with other providers, and the memberships that it holds with certain bodies. We capture associations such as the service provider being a partner of another provider, being a subsidiary, an owner, a supplier to, an agency, a division and a branch. We objectify this relationship to form a new entity called "AssociatedProvider". The intent of the objectification is to allow us to attach the fact type "has association that was certified by". This allows a provider to state that their association with another provider is confirmed by another party.

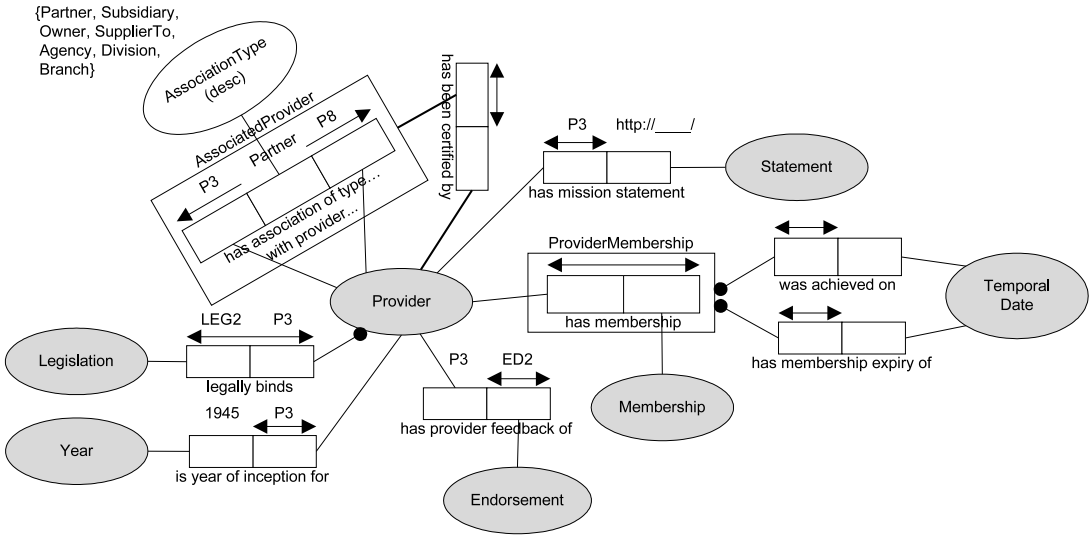


Figure 6.2: Provider related trust

The non-functional properties of trust that we directly represent with a service include two previously mentioned with respect to providers (the year of inception and links to endorsements received), and whether the payment obligation for a service can be executed using a particular escrow or insurance service [see Figure 6.3]. Specific mention is made of two types of services - escrow and insurance. We believe the prevalence of these two generic services warrant their inclusion as a property within our models. Finally, we provide for the capturing of a service provider who has verified the description. This is distinct from the conformance to a particular standard that is discussed in section 6.2.1.

The cumulative result of both our direct and indirect models with respect to trust is a domain independent approach that allows service requestors to determine the non-functional properties of services that best fulfil their requirements for trust. The following example query filters instances of directly modelled trust for a service

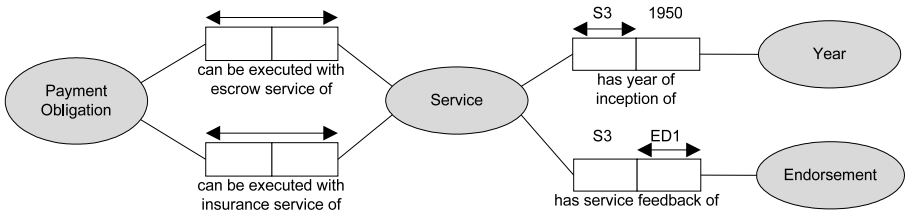


Figure 6.3: Services related trust

provider and a service based on the following criteria: the service inception is prior to 1990, the service provider inception is prior to 1985, and the provider is a partner with "Microsoft".

Provider
- ⊢ has year of inception < 1985
- ⊢ has association of type "Partner" with Provider
  - ⊢ has provider name "Microsoft"
- ⊢ offers Service
  - ⊢ has year of inception < 1990

The inverse of trust expressed so far (i.e. the trust of the service provider in the service requestor) is also be indirectly represented in the models. For example, the payment obligation model [see Figure 4.4] allows a service provider to request that a "deposit" (or bond) can be paid by the requestor. This type of payment increases the service provider's trust in a requestor as it represents a form of commitment on behalf of the service requestor.

6.2 Quality

Representing quality of service is difficult whilst trying to maintain a domain-independent view of service description. We take the view that service providers might prefer to capture service quality with respect to a standard, an industry benchmark and/or a ranking scheme. The latter two approaches also allow a provider to state a comparative assessment of their service with respect to an industry benchmark or ranking scheme, whether it be a self-assessment or an independently verified assessment of their conformance. We differentiate between industry benchmarks and rating schemes in the following manner. We consider an industry benchmark to be similar to a survey of service providers. These surveys capture certain indicators about the service. For example, a hotel industry benchmark may capture an indicator such as revenue per available room or average room rate per night. Ranking schemes we consider are a form of ranking of a service. For example, restaurants can be ranked in the Michelin Guide [63] according to the number of Michelin stars that they have received (between 1 and 3).

We have used existing work [124] as the basis for determining the aspects of service quality. They outline eleven (11) dimensions of perceived electronic service quality. We subscribe to Zeithaml's notions of reliability, responsiveness, access, efficiency, assurance/trust, security/privacy, price knowledge, customisation and believe that they can be captured using the models that we are about to present, or are dealt with in detail in other models. We consider two of the dimensions of ease of navigation and site aesthetics to be specific to electronic services. As we are trying to capture both electronic and traditional services we have ignored these dimensions.

Expectation of quality is importantly distinguished in [58] as occurring before the service interaction. We provide support for the aforementioned dimensions by including their description within our models. The expectation of service quality is therefore based on the description that is provided to the service requestor by the service catalogue. Whilst all the dimensions encourage expectations, the final dimension of flexibility appears to be dependent on some of the non-functional properties such as the ways to pay, buy or return items. It also includes notions such as the way to search for items. We believe that a service that offers sufficient flexibility to a service requestor in terms of the options available to them will determine a requestor's view of quality with respect to flexibility.

6.2.1 Standard, benchmark and ranking schemes

We use the term standard to refer to items "established or widely recognized as a model of authority or excellence" [80] [see Figure 6.4]. Standards are commonly published by service industries (captured in our model using a UN/SPSC code [105]) or service provider(s). They are regularly referred to using a title name and a publication date. Standards also have one or more author names, a version number, and a status (reflecting whether they are in a draft or final stage of completion). We provide a link to a location where the standard may be accessed, a reference to a standard superseding another standard, and that a standard offers differing levels of conformance. This allows us later to refer to the comparison of the service provider to the criterion within the standard using the standard level. We provide the means of nominating one or more regions as being covered by the standard. This approach permits us to store the standard that the service or provider complies with. In some cases standards have a functional focus (e.g. WSDL). With this type of standard it is important to also capture the location of the specific WSDL description for the service. We consider the reference to the WSDL to be captured by the functional description of the service (i.e. associated with the description of the service capability). As previously outlined this thesis relates to non-functional properties.

We further subtype standards into assessment standards. As previously mentioned this allows us to state a comparative assessment with respect to a standard for a service provider. Industry benchmarks are analogous to a survey, whilst we consider ranking schemes to be a form of ranking for a service [see Figure 6.5].

We consider benchmarks to have one or more indicators. An indicator can be thought of as a key performance measures or similar, that tracks a particular item

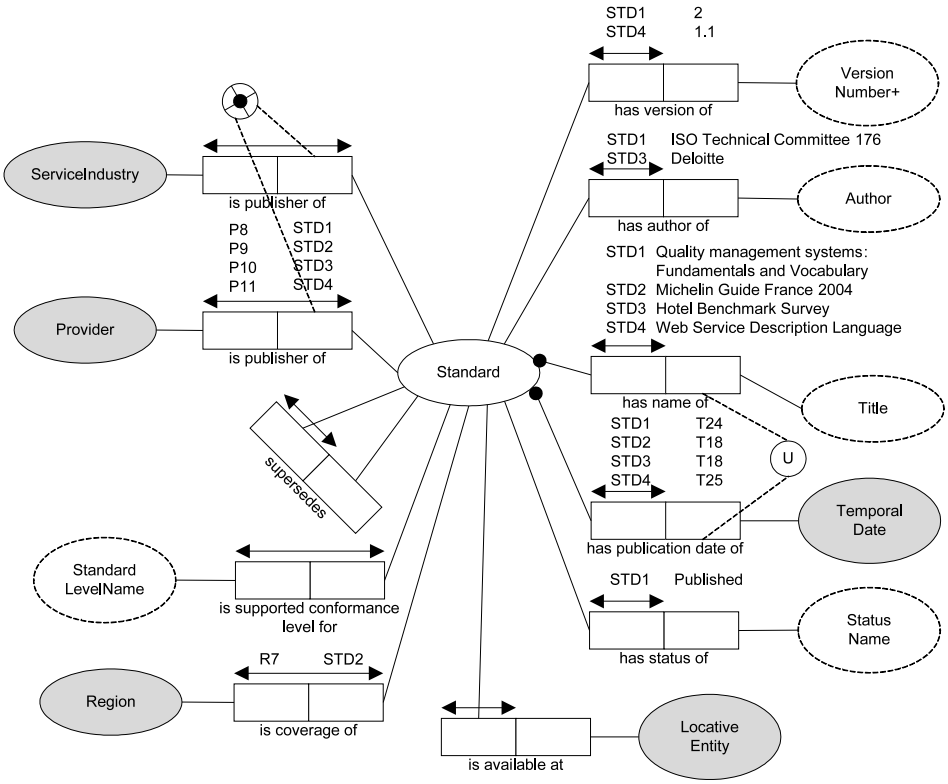


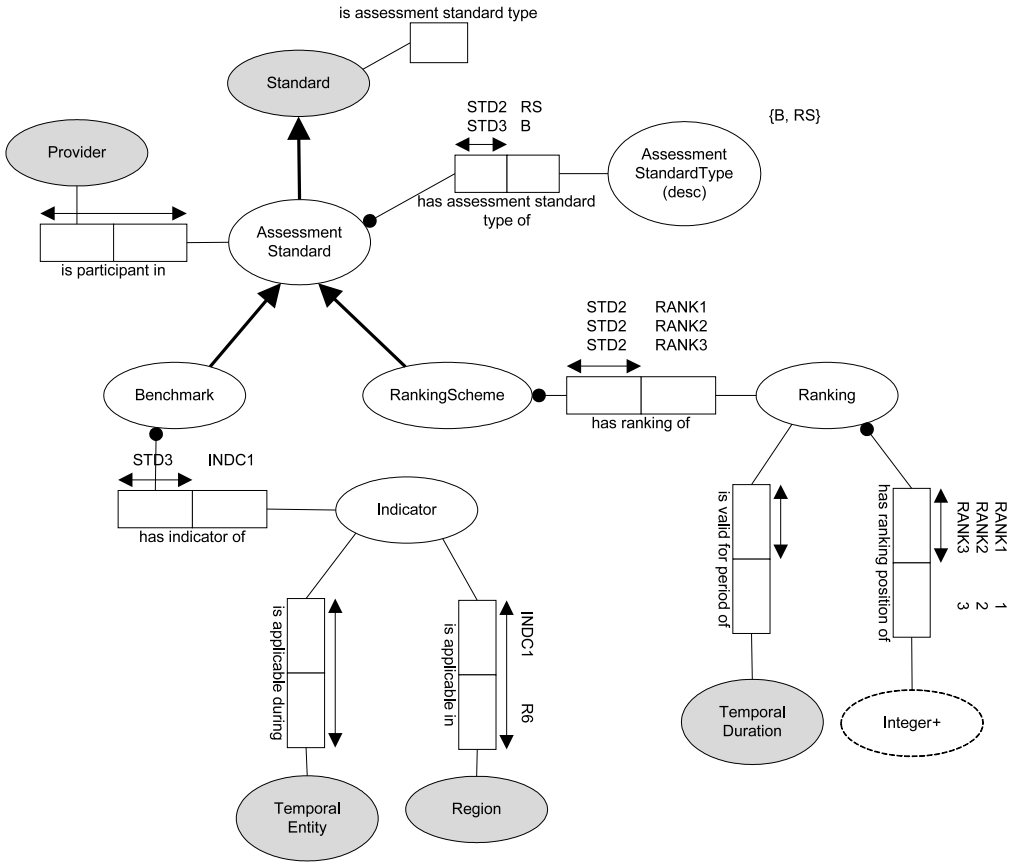
Figure 6.4: Standards

(e.g. average room rate per night). Indicators may apply during a certain temporal period or duration, may be valid for a particular region (e.g. a state within a country), and percentage change from the previous indicator result. Ranking schemes have one or more rankings that are normally valid for a specific period (e.g. 12 months) and have a ranking position (e.g. we refer to this using an integer value).

We have not expressed the specific information about the indicator or the ranking. We have subtyped these two entities from a super-type that we refer to as a rating [see Figure 6.6]. We consider ratings to have a name (the item that we are referring to) and a description. Ratings have a rating value that is an expression of the rating as either a percentage, a numeric value, a price, or a string representation.

Having shown how we capture the information relating to standards we can now express a comparative assessment with respect to the standard for the service [see Figure 6.7]. We provide for assessments for three quality dimensions: reliability, responsiveness and efficiency of the service. The service can capture the assessment of a particular dimension using a rating value of a particular rating with an assessment standard. Additionally, we capture that this comparative assessment was independently verified by another provider, as well as the date it was achieved.

In addition to a comparative assessment, we consider that quality of a service may require some form of feedback endorsement for a particular dimension of the service, that it allows configuration of a user profile at a certain location, and that it captures an interaction history. Providers may also achieve conformance of a



each AssessmentStandard **is a** Standard **that** is assessment standard type
each Benchmark **is a** AssessmentStandard **that** is of AssessmentStandardType 'B'
each RankingScheme **is a** AssessmentStandard **that** is of AssessmentStandardType 'RS'

Figure 6.5: Benchmarks

standard for a service on a particular date. This compliance has a conformance rating.

So this has enabled us to specify quality with respect to a service. In Figure 6.8 we capture quality with respect to a provider. This type of compliance defines standards that are met by the provider (e.g. ISO 9001).

6.3 Security

We attach the non-functional property of security to the locative aspects of a service (i.e. where it can be requested, provided, paid for, queried for further information etc). This is due to our belief that interactions that occur between the service provider and the service requestor must occur at a location or over some form of communications medium. We include the communications medium within this definition as there may be a distance between the service provider and the service requestor (e.g. a web service request). With neither of these dimensions do we attempt to describe the security of the service in-depth. We prefer to provide an

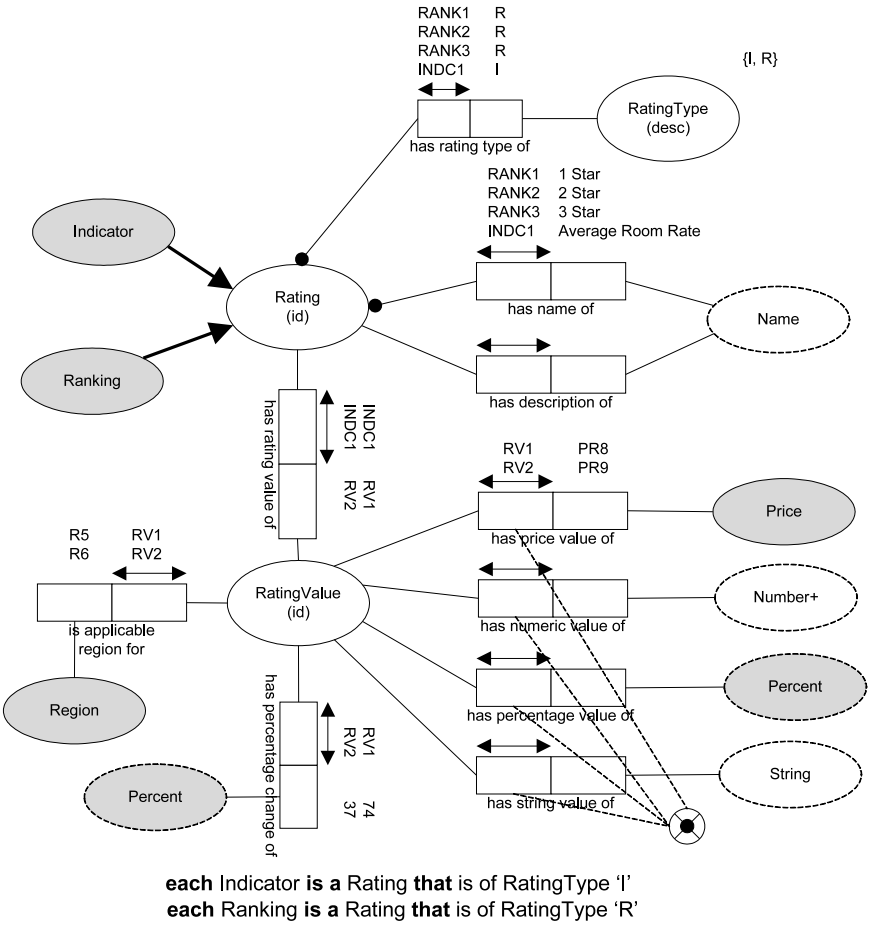


Figure 6.6: Ratings

overview of the type of security that the service requestor can expect.

We divide our discussion of security into two dimensions: identification and confidentiality. Examples of identification requirements for services include:

- Automated Teller Machine (ATM): Access to an ATM normally occurs using a card and a personal identification number (PIN). This can be considered the "something you have" and the "something you know" principles that together provide sufficient information for the ATM to determine if you have the permission to access the ATM.

- Web Site Access: Some web sites are secured with a username (or customer number) and password combination, possibly just a password. Examples include online banking, newspaper web sites, and web-based email.

- Biometric: Some services require biometric related information to determine if a requestor has access to a service. Biometric information is categorised as appearance, social, natural, biodynamics and imposed characteristics [25]. Common biometric security uses finger print, retinal scan and facial scan technologies.

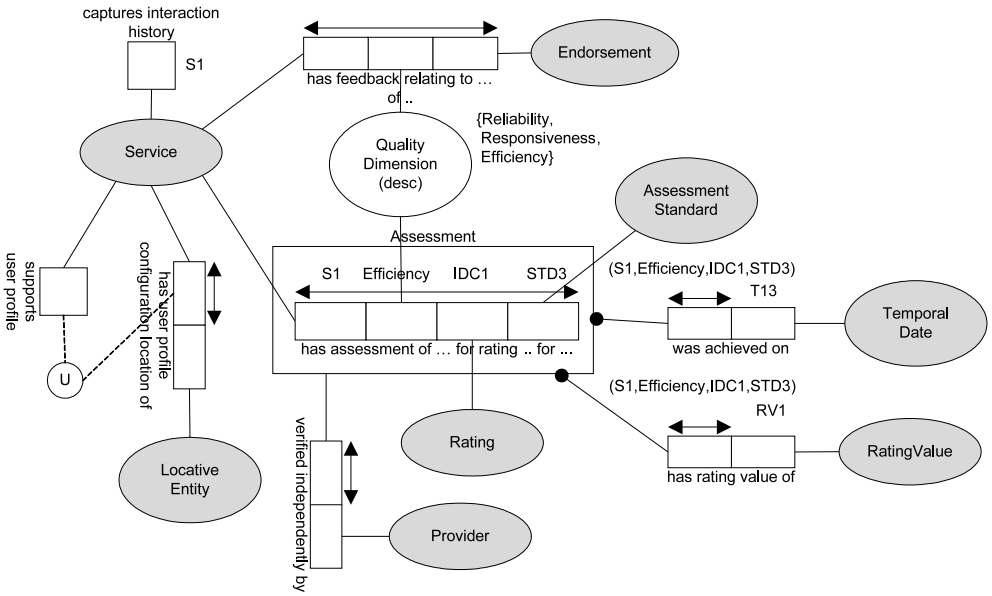


Figure 6.7: Service quality

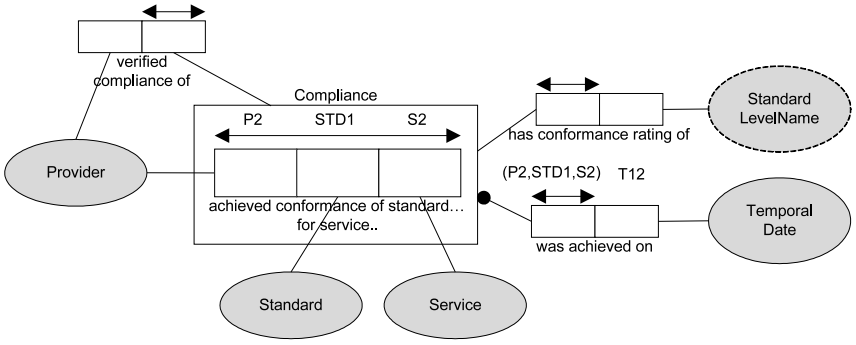


Figure 6.8: Provider quality

We present our model for security identification in Figure 6.9. We stipulate that a location may have one or more identification requirements. The identification requirement will be composed of a mandatory set of identification types that can be presented by the service requestor, as well as additional identification item. It is possible for the service provider to allocate a number of identification points per identification requirement/identification type combination (referred to in the model as AcceptableIdentification), as well as stipulating that the identification requirement must meet a collective points total to be achieved.

We consider identification types to include (but not be limited to): PIN, password, username, birth certificate, marriage certificate, membership card, licence, passport, student card, physical key, X509 certificate, Kerberos ticket, debit card, credit card, or a national identity card. Personal information such as names, addresses, dates of birth, or phone numbers are sometimes used to validate claims for access. In a WS-Security context this is referred to as proof of possession, and is used to validate unendorsed claims [65]. An endorsed claim is backed by the presen-

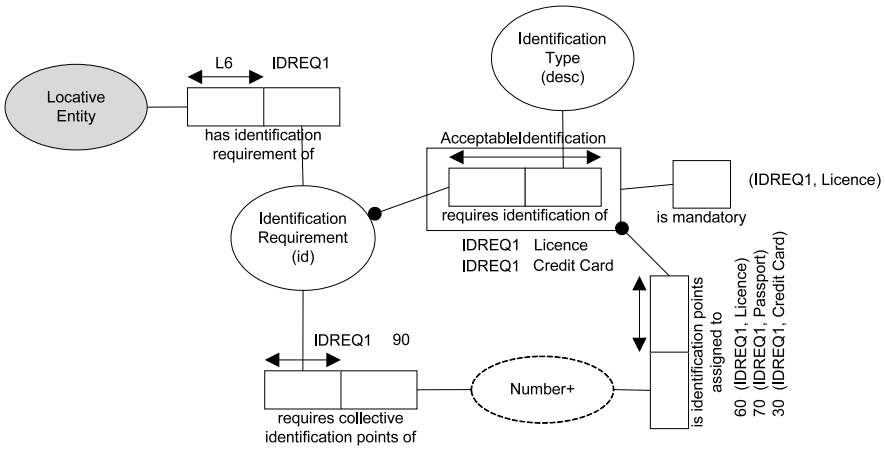


Figure 6.9: Identification

tation of an X509 certificate or a Kerberos ticket. These identification types have been endorsed by another party (e.g. Verisign). Other identification types include biometric based mechanisms: facial scan, retinal scan, iris scan, thumbprint, voice, finger print set, hand geometry and DNA. A table similar to the various characteristics of payment instruments could be constructed for these identification types.

The second dimension of security that we model is confidentiality [see Figure 6.10]. Confidentiality has two connotations depending on the domain. Firstly, it can be considered as the authorising of access or as an ethical principle that must be adhered to (e.g. in fields like medicine or journalism). We consider that this type of confidentiality is normally formalised into a confidentiality agreement. These types of agreements normally have a temporal interval that defines when they are applicable, and a legal jurisdiction within which it is controlled. Finally, we consider that confidentiality is applicable to the communications mechanism over which a service provider and service requestor interactions occurs. We capture the name, the key length and provide a link to the standard that defines the details of the encryption technique (or algorithm).

Summary

This chapter has presented a discussion of trust, quality and security. These non-functional properties have a significant impact on the perception that a service requestor has of the service provider or the service. We have represented these properties both indirectly within our models (particularly trust) and directly. Of particular interest is our treatment of quality. Our approach to quality allows a service provider to compare themselves, or their service to an existing standard, benchmark, and/or ranking scheme. We believe that this approach results in service providers agreeing on common standards, benchmarks or ranking schemes for particular domains or industries.

The next chapter presents the substantiation of our research in a number of contexts. We primarily undertake this substantiation to highlight adherence of our

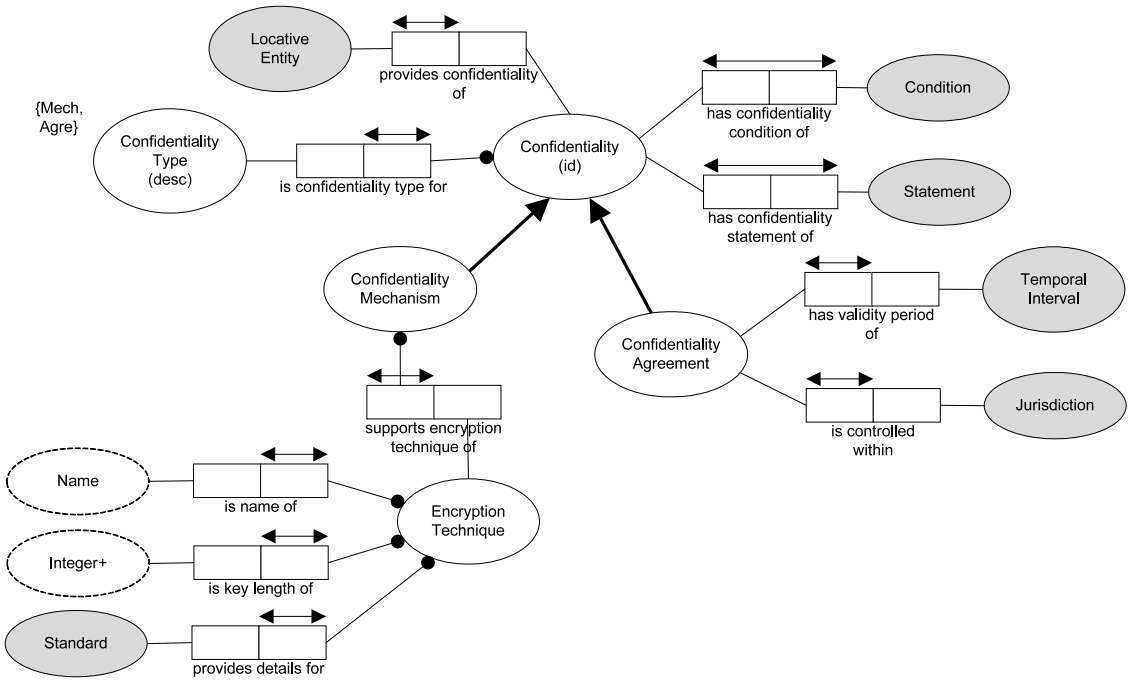


Figure 6.10: Confidentiality

work to the criteria for a solution that were presented in chapter 2 of this thesis. It also provides a glimpse of the depth of the semantic richness of our models, as well as an indication of the number of opportunities that we see for usage of the models.

Chapter 7

Model substantiation

The model presented within this thesis provides an abstract language for describing a range of domain independent non-functional properties of services. We have categorised these into the areas of availability (temporal and locative), payment, price, security, trust, quality, discounts, penalties, rights and obligations. In this chapter we attempt to substantiate the model by using it in different ways. The first of these contexts involve the use of our non-functional properties as a concrete representation within a web service interaction, and subsequently as a semantic annotation to a web service description. The second context involves a discussion of our non-functional properties in electronic tendering.

These contexts are intended to highlight adherence to some of the criteria for a solution that were presented in Chapter 2 of the thesis. The criteria that we believe are displayed within this chapter are expressiveness, executability, comprehensibility, and suitability. The reason for partial adherence here is that some criteria apply to the conceptual model, not the more physical (or concrete) perspective of these substantiations. It should also be noted that some of these criteria are not mutually exclusive to the conceptual or physical perspectives.

7.1 Illustrative support

The basis for the first of the contexts is our fabricated service provider - "Just Park It" (JPI). JPI provides car parking services to clients at ten locations within a bustling city of approximately 1 million people. Over a 15 year period JPI has become increasingly sophisticated in the provision of the services that it provides. We believe that JPI's organic growth in sophistication is indicative of numerous service providers across many domains. In Figure 7.1 we present this spectrum of service provision sophistication.

We believe that the spectrum includes:

- Bricks 'n' mortar service - At its inception JPI was a traditional bricks 'n' mortar service. Service requestors would arrive at the car park building and would receive a car park ticket on entering the facility. Should the car park be full, the service requestor would try another JPI location, or that of a

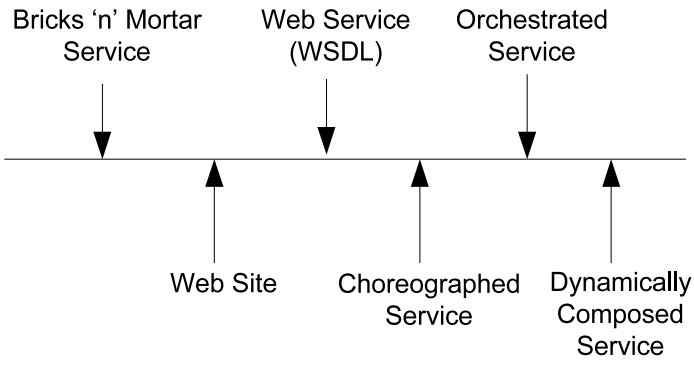


Figure 7.1: Spectrum of service provision sophistication

competitor. JPI used traditional forms of advertising such as handing out pamphlets, print media (i.e. newspaper, magazines) and sometimes television. Service requestors were required to deal with JPI in a manual manner. Discovery of service related non-functional properties such as price was performed by requestors using signs at the entry to the building.

- Web site - As the Internet gained in popularity JPI combined its traditional bricks 'n' mortar services with that of a web site. This was primarily to increase visibility of the services being provided. Some level of forms were available to requestors to enable them to contact JPI and to provide feedback.

- Web service (WSDL) - Shortly after the arrival of Web Service standards (WSDl, SOAP and XML) JPI received a request from a local rental car company to create a web service. The rental car company was setting up an office in the city to allow requestors to return cars there. Rather than take out permanent space in the city they decided that it might be more economical to only book a car park when cars were to be returned to that office. JPI created their web service for the rental car company but this web service was also able to be used by other service providers. The rental car company negotiated a better rates schedule manually with JPI.

- Choreographed service - JPI went through a process of rewriting their car park tracking system. A service oriented architecture was chosen to allow for more business agility in the future. Developers of the new system utilised the existing web service as part of the development.

- Orchestrated service - JPI believe that to increase the future utilisation of their car parks they should enable business partners to include their service within the service of a business partner. For example, travel agencies could provide a web site that allows a traveller to book their airfare, accommodation, a car to get from the airport to the hotel, and a car park at a nearby car parking station. This is where JPI is utilised. Once building the itinerary of the traveller, the travel agency could call the respective web services of the airline, hotel, car rental agency and JPI.

- Dynamically composed service - With sufficiently well described web services, JPI believes that it would be well positioned to be dynamically discovered (via the publishing of its rich service descriptions) and included within a service orchestration.

7.1.1 Why use this example?

There are a number of reasons why we have chosen to utilise this car park service as our example. Our models are derived from an analysis of a great many existing services across numerous domains. We feel that a car park service is sufficiently common to aid with understanding, as well as highlighting the following:

- Our models distinguish between the request and provision of the service. This distinction allows us to describe services (such as in this example) that have an electronic request mechanism (e.g. web services) and a manual provision. We prefer not to limit the scope of our non-functional properties to either web services, or conventional services.

- We believe that this example shows that common, everyday services require semantic richness for their service descriptions. Service providers require an ability to describe services naturally - for example, by allowing recurring temporal intervals (such as Monday to Friday 9am to 5pm) to be described. This example allows us to show this type of expressiveness for the service providers during the description process.

7.2 Prerequisite work

The models developed and presented within this thesis have largely been based on descriptions of everyday services. Descriptions that we have previously outlined as being sourced from newspaper, television and magazine advertisements, pamphlets, corporate web sites etc. Having analysed these services we developed the ORM models to generate an abstract language for describing the domain independent non-functional properties of services. To verify the validity of the models we applied populations (essentially examples) to the ORM models. Then, in an effort to substantiate the models we chose to convert it from its conceptual form (in Object Role Modelling notation) to a concrete representation, specifically XML Schema. The primary reason for converting the ORM notation to XML Schema was due to WSDL's support for XML Schema types, and the availability of a mapping technique between the two forms. This usage in WSDL is outlined in more detail in section 7.3.

With the XML Schema created, we have then been able to generate rudimentary user interfaces as a means of visualising the models. Finally, the contexts presented within this chapter are a final means of testing the robustness and applicability of our work. At each of the visualisation and contextual steps we have reviewed the outputs and updated the original ORM models. This conversion process is depicted in Figure 7.2.

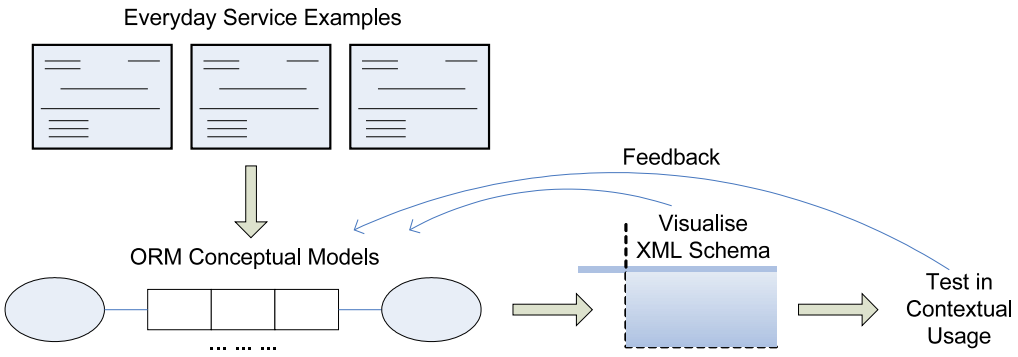


Figure 7.2: Conversion process

Conversion of ORM models to XML Schema

The process of converting from ORM notation to XML Schema was performed using the three step process and the twelve guiding rules defined in [11]. A detailed discussion of the ORM notation has previously been provided in Chapter 2. The steps for the specific conversion from ORM to XML Schema are as follows:

- Generate a type definition for each ORM object type.
- Build a complex type definition for each major fact type grouping.
- Create a root element for the whole schema and add keys and key references.

The resultant XML Schema provides a language for communicating non-functional properties.

7.2.1 A walkthrough of the conversion process

As a means of assisting the reader with understanding the conversion process, we now provide a walkthrough of a concrete example. The advertisement (presented in Figure 7.3) is from our illustrative service provider, JPI. The advertisement is indicative of a simple pamphlet that is handed out at locations around JPI car parking stations. This particular advertisement is for one specific location.

We are focusing on the conversion of one part of the advertisement, the location of the car parking station (see the dotted oval). The locative information in this advertisement has been specified using a street address. The LocativeEntity ORM model outlined in Chapter 3 provides a set of sub-types, of which one is Address. It in turn has two subtypes - StreetAddress and PostboxAddress. We repeat most of the models within this chapter that are necessary to capture the locative details presented in the advertisement. The repeated models have however been updated to include JPI specific ORM populations.

The conversion process begins with the identification of a service description (e.g. this advertisement). We analysed these descriptions to determine an ORM model that best represents it. To help us determine if our ORM model is valid we apply population constraints as a means of checking its validity. This aspect of the

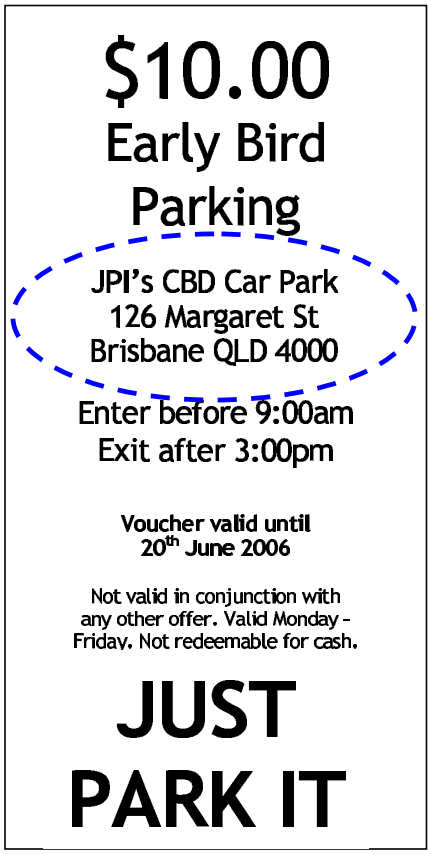


Figure 7.3: Car park advertisement

conversion process is outlined in the paragraphs that follow. We have assumed that our ORM models exist prior to identification of the service description (i.e. we are not walking through the analytical process of developing our ORM models).

Applying populations to the ORM models

The first model that we present (Figure 7.4) outlines the LocativeEntity subtype referred to as an "Address". This subtype captures the properties that are common to two different types of addresses supported in our models - StreetAddress and PostboxAddress. These common properties include items such as city, country and postcode. We assigned the surrogate identifier "L-JPI" to the street address ("JPI'S CBD Car Park, 126 Margaret St, Brisbane QLD 4000") in our ORM model. The applicable fact types within the "Address" model are as follows:

- has state/territory of
- has city of
- has postcode/zipcode of

Two of these fact types refer to regions - QLD (for state/territory) and Brisbane (for city). We have assigned surrogate identifiers for each of the regions (R-Q and R-

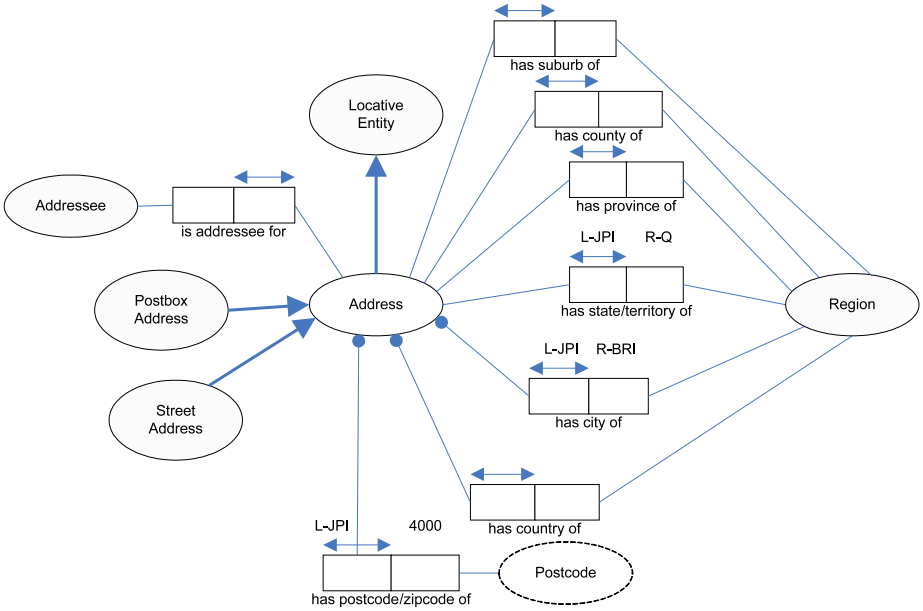


Figure 7.4: Car park populated - Locative address

BRI respectively). We have not included the Locative ORM model with respect to regions here as we feel that it complicates this explanation. It can be assumed that the definition for these regions is available through its surrogate identifier reference.

At this point, our populated "Address" ORM model only expresses some of the locative information contained within the advertisement. Part of the remaining information is specific to street addresses and is contained in Figure 7.5. The fact types that we utilise within the "StreetAddress" ORM model are as follows:

- has street number of
- has street type of
- has street name of

The final piece of information to be represented from the advertisement is the common name - "JPI's CBD Car Park". We provide the ability for all LocativeEntities to have a common name associated with it. This has been depicted in Figure 7.6. We could further restrict the use of this common name to a particular LocativeEntity (e.g. a city like Brisbane) but we have chosen not to do so for simplicity. The reader may also notice that the subtype defining role is also populated to denote that the LocativeEntity in question is a StreetAddress (or "SA" as our shorted enumeration constraint is defined).With this information populated on the ORM models we have outlined the first step in the conversion process.

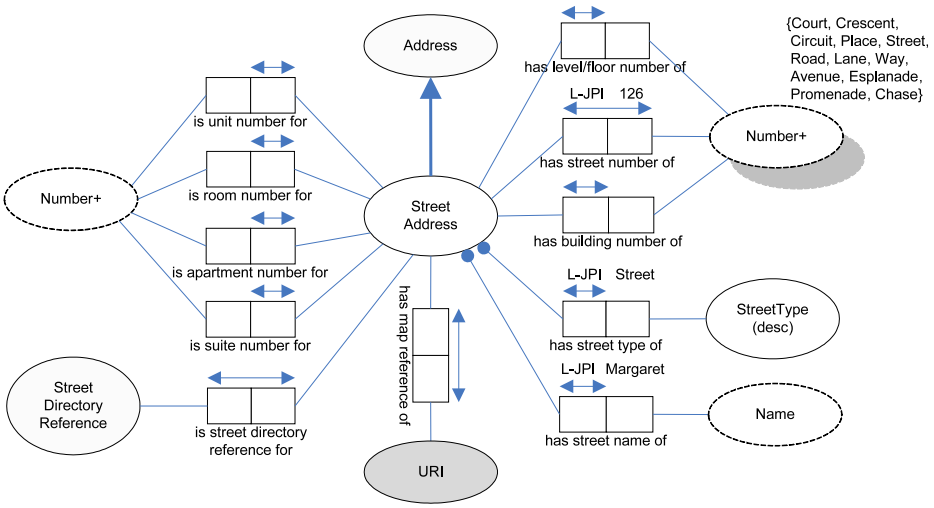


Figure 7.5: Car park advertisement - Locative street address

Transitioning ORM to XML Schema

An example of the converted ORM model to XML Schema is provided for the LocativeEntity supertype. The LocativeEntity ORM object type is outlined in Chapter 3, and additional locative properties are available in Chapter 6. See Figures 3.9, 3.28 and 6.9 for our locative non-functional properties. Note however that this is distinct from their use in defining locative availability for request or provision.

Step 1 - Create a type definition for each ORM object type Using the steps identified in [11] we begin by generating a type definition for each ORM object type. This includes types such as "StreetType" (see Listing 7.1) which represents an enumeration of the different street types. Note however that this list is intended to be indicative, not complete.

```
<xsd:simpleType name="StreetType">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="Circuit"/>
    <xsd:enumeration value="Place"/>
    <xsd:enumeration value="Street"/>
    <xsd:enumeration value="Road"/>
    <xsd:enumeration value="Lane"/>
    <xsd:enumeration value="Way"/>
    <xsd:enumeration value="Avenue"/>
    <xsd:enumeration value="Esplanade"/>
    <xsd:enumeration value="Promenade"/>
    <xsd:enumeration value="Chase"/>
  </xsd:restriction>
</xsd:simpleType>
```

Listing 7.1: Type definitions

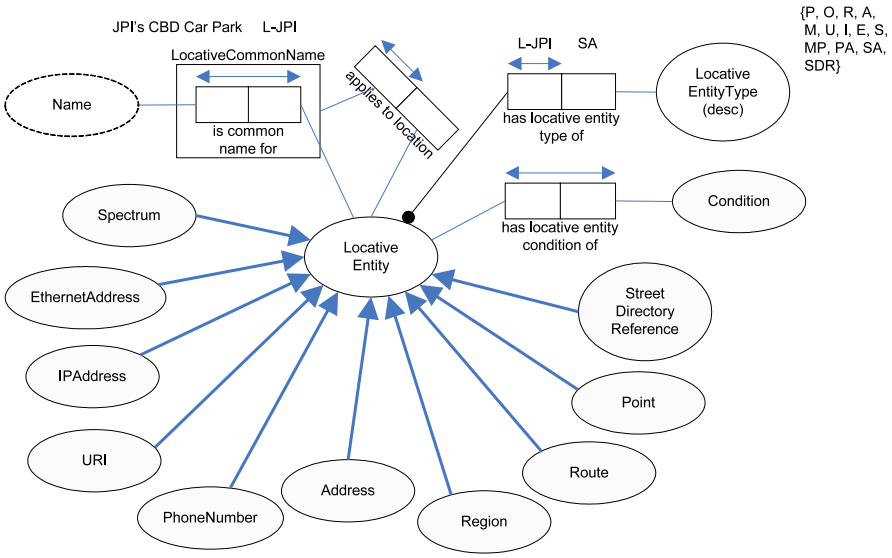


Figure 7.6: Car park advertisement - Locative subtypes

Step 2 - Build a complex type definition for each major fact type grouping The second step involves building a complex type definition for each major fact type grouping. We highlight in our example three of the object types of interest - LocativeEntity, Address and StreetAddress. Listing 7.2 provides us with an outline of the possible locative subtypes. To enable ease of understanding in the XML Schema we have named our element StreetAddress and its type definition L_StreetAddress. This naming convention assisted with the user interfaces that were generated when we conducted the next step in the conversion process, the visualisation step. The namespace outlined in the type definitions (i.e. nfp) provides a qualifying mechanism to identify the types that we have developed.

```
<xsd:complexType name="LocativeEntity">
  <xsd:sequence>
    <xsd:element name="hasLocativeValue">
      <xsd:complexType>
        <xsd:choice>
          <xsd:element name="Point" type="nfp:L_Point"/>
          <xsd:element name="MovingPoint" type="nfp:L_MovingPoint"/>
          <xsd:element name="Route" type="nfp:L_Route"/>
          <xsd:element name="Region" type="nfp:L_Region"/>
          <xsd:element name="Address" type="nfp:L_Address"/>
          <xsd:element name="StreetAddress"
             type="nfp:L_StreetAddress"/>
          <xsd:element name="PostboxAddress"
             type="nfp:L_PostboxAddress"/>
          <xsd:element name="URI" type="nfp:L_URI"/>
          <xsd:element name="PhoneNumber" type="nfp:L_PhoneNumber"/>
          <xsd:element name="IPAddress" type="nfp:L_IPAddress"/>
          <xsd:element name="EthernetAddress"
             type="nfp:L_EthernetAddress"/>
          <xsd:element name="Spectrum" type="nfp:L_Spectrum"/>
          <xsd:element name="StreetDirectoryReference"
```

```
            type="nfp:L_StreetDirectoryReference"/>
        </xsd:choice>
      </xsd:complexType>
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>
```

Listing 7.2: Locative subtypes

As a means of providing all the types identified in Listing 7.2 with the common properties of all LocativeEntities we have created a L_LocativeEntity type that is presented in Listing 7.3. This entity provides support for fact types such as has a common name, supports written and spoken language, that it may be communicated with according to a standard, has identification requirements and provides a level of confidentiality. Using the "maxOccurs" and "minOccurs" attributes of the elements we are able to provide constraints that apply within the ORM model. There are a range of LocativeEntity subtypes including Point, MovingPoint, Route, Region etc. For the sake of brevity and remaining specific to our example not all locative subtypes have corresponding listings in this chapter. All subtypes are however outlined in Appendix B.

The populations applied to the ORM model in Figure 7.6 provide reference to the subtype defining role (i.e. has locative entity type of) and the nested fact type for a locative common name. To represent the locative common name we use a combination of both the common name and optionally, the region that the common name is used in. The XML Schema for this common name type (L_LocativeCommon) is presented in Listing 7.4.

With the elements "supportsWrittenLanguageOf" and "supportsSpokenLanguageOf" we have imported a complex type from another namespace [97]. These codes have been previously defined as part of the ISO639-2 standard - Codes for the representation of names of languages–Part 2: Alpha-3 code.

```
<xsd:complexType name="L_LocativeEntity">
  <xsd:sequence>
    <xsd:element name="hasLocativeEntityConditionOf"
      type="xsd:anyURI"/>
    <xsd:element name="hasLocativeCommonNameOf"
      type="nfp:L_LocativeCommon" maxOccurs="unbounded"
      minOccurs="0"/>
    <xsd:element name="supportsWrittenLanguageOf"
      type="iso639-2:CodeType" maxOccurs="unbounded" minOccurs="0"/>
    <xsd:element name="supportsSpokenLanguageOf"
      type="iso639-2:CodeType" maxOccurs="unbounded" minOccurs="0"/>
    <xsd:element name="canBeCommunicatedWithAccordingTo"
      type="nfp:Q_Standard" maxOccurs="unbounded" minOccurs="0"/>
    <xsd:element name="hasIdentificationRequirement"
      type="nfp:IdentificationRequirement" maxOccurs="1"
      minOccurs="0"/>
    <xsd:element name="providesConfidentialityOf"
      type="nfp:Confidentiality" maxOccurs="1" minOccurs="0"/>
  </xsd:sequence>
```

```
</xsd:complexType>
```

Listing 7.3: Locative entity

In our population of Figure 7.6 we apply a locative common name of "JPI's CBD Car Park", in reference to the street address details provided in the advertisement.

```
<xsd:complexType name="L_LocativeCommon">
   <xsd:sequence>
      <xsd:element name="hasLocativeCommonNameOf" type="xsd:string"
          maxOccurs="1" minOccurs="1"/>
      <xsd:element name="appliesToLocation"
          type="nfp:L_LocativeEntity" maxOccurs="1" minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Listing 7.4: Locative common name

In moving on to our next major fact type grouping, L_Address (see Listing 7.5) we can see that this complexType inherits all the properties of the supertype (L_L ocativeEntity) plus it extends it further to support attributes that are common to all addresses (e.g. city, country). We optionally support addresses that utilise surburbs, counties, provinces, states, territories and addresses. Note the `minOccurs="1"` that is utilised on the fact types "hasCountryOf" and "hasPostcodeOf" which enforces the requirement to supply this property. We do not outline the "L_Addressee" entity here. See Appendix B for more information.

```
<xsd:complexType name="L_Address">
  <xsd:complexContent>
    <xsd:extension base="nfp:L_LocativeEntity">
      <xsd:sequence>
        <xsd:element name="hasSuburbOf" type="nfp:L_Region"
           maxOccurs="1" minOccurs="0"/>
        <xsd:element name="hasCountyOf" type="nfp:L_Region"
           maxOccurs="1" minOccurs="0"/>
        <xsd:element name="hasProvinceOf" type="nfp:L_Region"
           maxOccurs="1" minOccurs="0"/>
        <xsd:element name="hasStateOf" type="nfp:L_Region"
           maxOccurs="1" minOccurs="0"/>
        <xsd:element name="hasTerritoryOf" type="nfp:L_Region"
           maxOccurs="1" minOccurs="0"/>
        <xsd:element name="hasCityOf" type="nfp:L_Region"
           maxOccurs="1" minOccurs="1"/>
        <xsd:element name="hasCountryOf" type="nfp:L_Region"
           maxOccurs="1" minOccurs="1"/>
        <xsd:element name="hasPostcodeOf" type="nfp:L_Postcode"
           maxOccurs="1" minOccurs="1"/>
        <xsd:element name="hasAddresseeOf" type="nfp:L_Addressee"
           maxOccurs="1" minOccurs="0"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

Listing 7.5: Locative subtype - Address

Listing 7.6 extends the "Address" fact type created previously to support the properties specific to street addresses. From the XML Schema we can see that it extends the L_Address complex type which in turn extends the L_LocativeEntity complex type. The "hasStreetType" fact type refers to the simple type created in step 1 of the ORM to XML schema approach (see Listing 7.1). We do not provide specific details here about the L_StreetDirectoryReference complex type. We believe it is sufficient to say that for a particular street address it is sometimes useful to also store a street directory reference.

```
<xsd:complexType name="L_StreetAddress">
  <xsd:complexContent>
    <xsd:extension base="nfp:L_Address">
      <xsd:sequence>
        <xsd:element name="hasStreetType" type="nfp:StreetType"/>
        <xsd:element name="hasStreetNameOf" type="xsd:string"/>
        <xsd:element name="hasStreetNumberOf"
           type="xsd:nonNegativeInteger"/>
        <xsd:element name="hasBuildingNumberOf"
           type="xsd:nonNegativeInteger"/>
        <xsd:element name="hasLevelOf" type="xsd:nonNegativeInteger"/>
        <xsd:element name="hasUnitNumberOf"
           type="xsd:nonNegativeInteger"/>
        <xsd:element name="hasRoomNumberOf"
           type="xsd:nonNegativeInteger"/>
        <xsd:element name="hasApartmentNumberOf"
           type="xsd:nonNegativeInteger"/>
        <xsd:element name="hasSuiteNumberOf"
           type="xsd:nonNegativeInteger"/>
        <xsd:element name="hasStreetDirectoryReferenceOf"
           type="nfp:L_StreetDirectoryReference"/>
        <xsd:element name="hasProximityOf" type="nfp:L_Proximity"/>
        <xsd:element name="hasMapReferenceOf" type="xsd:anyURI"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

Listing 7.6: Locative subtype - Street address

Step 3 - Create a root element for the whole schema and add keys and key references Since our models are larger than the notion of locative entities, we have not used a root element that relates to location. Listing 7.7 provides the root node for our XML Schema. We believe that services have locative availability (for request and provision) which can be stated using our LocativeEntity XML Schema complex type.

```
  <xsd:element name="Service" type="nfp:S_Service"/>
```

Listing 7.7: Root node

Limited value in our work is realised by adding of keys and key references as per this step in the process. Our preference is to use an "id" attribute to identify all

schema components. Whilst this has not been added, it could easily be included and would take the form of:

```
<xsd:complexType name="L_StreetAddress"
                 id="nfp:locative:streetaddress">
   ...
</xsd:complexType>
```

Listing 7.8: Addition of id attribute

Visualising the XML Schema

Having converted the ORM model to XML Schema we utilised a visualisation tool to assist with our substantiation [120]. The result of visualising the XML Schema presented previously within this section is depicted in Figure 7.7. It should be noted that this is a partial visualisation. The screen itself has more properties (such as those inherited from the L_LocativeEntity and L_Address complex types) but due to space restrictions we have only presented the properties from the L_StreetAddress complex type.

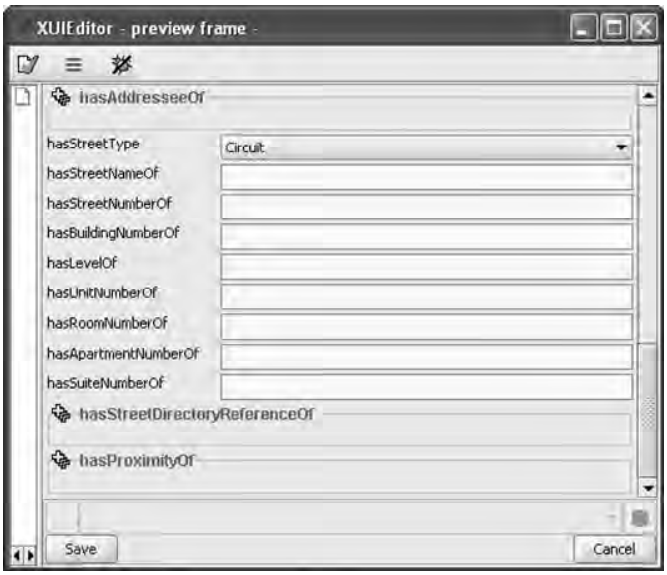


Figure 7.7: Visualisation of the L_StreetAddress XML Schema complex type

We are now able to utilise the XML Schema within the contexts described at the start of this chapter. The first of these contexts involve the use of our non-functional properties as a concrete representation within a web service interaction, and subsequently as a semantic annotation to a web service description.

7.3 A language for non-functional properties

With an XML Schema representation of the ORM models we are now able to communicate with other parties with respect to these non-functional properties. We

display this communication via the invocation of a Web service. The XML Schema representation allows us to use the non-functional properties as the input and/or output types of parameters that relate to a specific Web service operation.

Examples of the types are presented below. We developed a simple Web service that allows a service requestor to book a place in a JPI car park. The request for the Web service has a type called "makeBooking". In our example Web service we have defined the following non-functional properties as input parameters (Street Address, Anchored Point in Time and Discount). We have from our model as input to the operation. Listing 7.9 provides two important references within this context. WSDL provides for six major elements, the first of which involves types. The type section partially depicted in Listing 7.9 provides definitions for data types to be passed during calls to operations. Firstly, line 7 imports the namespace and therefore binds the namespace within a document to a location (i.e. the URI specified). Secondly, lines 11 - 16 provide parameters to the input message of our web service. Each of these input parameters is defined using the complex types from our XML Schema.

```
1  <types>
2    <s:schema elementFormDefault="qualified"
3       targetNamespace
4       ="http://www.service-description.com/CarParkBookingService"
5       xmlns:s="http://www.w3.org/2001/XMLSchema"
6       xmlns:nfp="http://www.service-description.com/NFP/"
7       xmlns:car
8       ="http://www.service-description.com/CarParkBookingService">
9    <s:import namespace="http://www.service-description.com/NFP/"/>
10   <s:element name="makeCarParkBooking">
11     <s:complexType>
12       <s:sequence>
13         <s:element name="streetAddress" type="nfp:L_StreetAddress"
14            minOccurs="0"/>
15         <s:element name="estimatedArrival"
16            type="nfp:T_AnchoredPointinTime" minOccurs="0"/>
17         <s:element name="estimatedDeparture"
18            type="nfp:T_AnchoredPointinTime" minOccurs="0"/>
19         <s:element name="discountToUse" type="nfp:D_Discount"
20            minOccurs="0"/>
21       </s:sequence>
22     </s:complexType>
23   </s:element>
```

Listing 7.9: WSDL types - Part 1

Listing 7.10, line 4 provides the definition of the sole element that represents the output from our Web service. The definition of its complex type is defined by the BookingReference complex type. Moving to line 12 we can see that the T_AbsolutePrice data type is used as part of the output to the Web service. In this case it provides a price estimate for the caller of the Web service.

```
1    <s:element name="makeCarParkBookingResponse">
2      <s:complexType>
3        <s:sequence>
4          <s:element name="makeCarParkBookingResult"
```

```
5             type="car:BookingReference" minOccurs="0"/>
6         </s:sequence>
7       </s:complexType>
8     </s:element>
9     <s:complexType name="BookingReference">
10      <s:sequence>
11        <s:element name="BookingReference" type="s:string"
12          minOccurs="0"/>
13        <s:element name="EstimatedCost" type="nfp:T_AbsolutePrice"
14          minOccurs="0"/>
15      </s:sequence>
16    </s:complexType>
17  </s:schema>
```

Listing 7.10: WSDL types - Part 2

The full details of the complexTypes that constitute the parameter types are also included in the WSDL. That is, types such as "nfp:L_StreetAddress" and "nfp:T_AnchoredPointinTime" are included. For brevity we have only depicted a partial description. It should be noted that the language is not specific to use within Web services. The XML Schema representation could be used where any interactions occur with respect to non-functional properties. For example, the YAWL workflow system [112] allows activities to be created with specific input and outputs. Typing of these inputs and outputs within YAWL can be provided as an XML Schema representation. The fact that this has been used within Web Services is just indicative of one type of interaction that the language can be used for.

7.3.1 Semantic annotation

In this context, we also provide semantic annotation of the input and output parameters that a Service Provider declares for their Web service operations. Providers may choose to use our non-functional language (i.e. the XML Schema types) in their service operations. Their preference may be to represent the non-functional properties differently. For example, giving a different name to the data types. They do however want to help requestors of the service understand what they mean by a particular parameter. To that end, semantic annotation assists the Service Provider state "when I mention or refer to this non-functional property on my service interface I am referring to the concept defined by $< PersonX >$".

Looking forward JPI have decided to provide additional semantics than just the functionally focused WSDL. By choosing to decide to use WSDL-S [1], JPI have the ability to add non-functional properties to the descriptions of their web service. WSDL-S provides "an extension attribute, namely modelReference, to specify the association between a WSDL entity and a concept in some semantic model. It can be added to a complex type, element, operation, as well as the extension elements - precondition and effect". We have depicted where this work is useful with respect to the spectrum of service provision sophistication (Figure 7.1) in Figure 7.8.

In our opinion that preconditions and effects are within the area of the functional description of a service. With this in mind, we took the following steps:

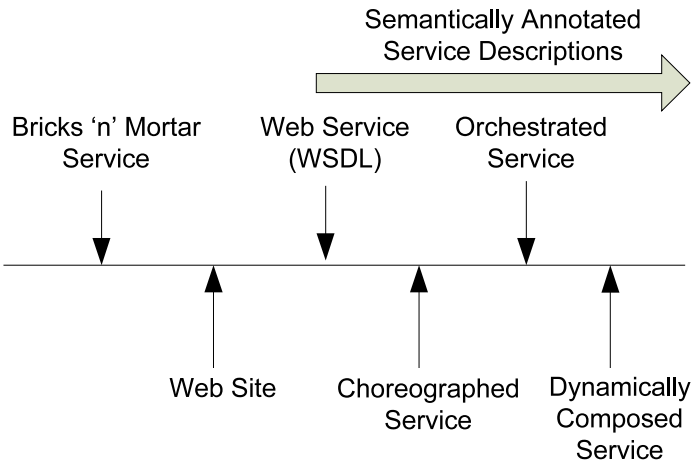


Figure 7.8: Semantic annotation

- We used the WSDL created in the previous context (see section 7.3) as the basis for this substantiation.
- We took the XML Schema and used two different converters ([81, 111]) to generate an OWL representation.
- We then used the Radiant: WSDL-S Annotation Tool [96] which is a plug-in to the Eclipse Integrated Development Environment. Radiant allowed the original WSDL file to be annotated with references to the OWL version of our semantic model.

These three steps were used in producing the WSDL-S file. Listing 7.11 introduces two namespaces (i.e. NFPOntology and wssem), where "NFPOntology" is our semantic model and "wssem" is the Web Service semantics model. This namespacing provides a method of qualifying the origin of an XML element or attribute by referring the namespace to a URI.

```
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:conv="http://www.openuri.org/2002/04/soap/conversation/"
    xmlns:cw="http://www.openuri.org/2002/04/wsdl/conversation/"
    xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
    xmlns:jms="http://www.openuri.org/2002/04/wsdl/jms/"
    xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
    xmlns:s="http://www.w3.org/2001/XMLSchema"
    xmlns:s0
    ="http://www.service-description.com/CarParkBookingService"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    targetNamespace
    ="http://www.service-description.com/CarParkBookingService"
    xmlns:NFPOntology
    ="http://www.service-description.com/NFP/owl/NFP-Full.owl#"
    xmlns:wssem="http://www.ibm.com/xmlns/WebServices/WSSemantics">
  <types>
```

Listing 7.11: WSDL-S - Part 1

Listing 7.12 provides an annotated version of the same WSDL presented in Listing 7.9. References such as `wssem:modelReference="NFPOntology#L_StreetAddress"` are links from elements within the WSDL to an external semantic OWL model. In this case, our semantic model generated using an XML Schema to OWL conversion tool.

```
<s:schema elementFormDefault="qualified"
  targetNamespace
  ="http://www.service-description.com/CarParkBookingService"
  xmlns:s="http://www.w3.org/2001/XMLSchema"
  xmlns:nfp="http://www.service-description.com/NFP/"
  xmlns:car="http://www.service-description.com/CarParkBookingService">
  <s:import namespace="http://www.service-description.com/NFP/"/>
  <s:element name="makeCarParkBooking">
    <s:complexType>
      <s:sequence>
        <s:element name="streetAddress" type="nfp:L_StreetAddress"
          wssem:modelReference="NFPOntology#L_StreetAddress"
          minOccurs="0"/>
        <s:element name="estimatedArrival"
          type="nfp:T_AnchoredPointinTime"
          wssem:modelReference="NFPOntology#T_AnchoredPointinTime"
          minOccurs="0"/>
        <s:element name="estimatedDeparture"
          type="nfp:T_AnchoredPointinTime"
          wssem:modelReference="NFPOntology#T_AnchoredPointinTime"
          minOccurs="0"/>
        <s:element name="discountsToUse" type="car:ArrayOfDDiscount"
          minOccurs="0"/>
      </s:sequence>
    </s:complexType>
  </s:element>
```

Listing 7.12: WSDL-S - Part 2

This approach allows us to also use WSDL-S's non-prescriptive extensions that support publishing into a UDDI registry. A side effect of this is that WSDL-S service descriptions can be discovered based on some non-functional properties and/or the usage of a particular semantic model that would be discovered via modelReferences within the WSDL-S file.

7.4 Tendering

Whilst existing web processes can be considered dynamic from a service requestor perspective, the service provider must advertise their service description into a catalogue. This results in a static description of a service that must be general enough to cater for all requestors undertaking service discovery. We feel that dynamic web processes should extend to the service descriptions advertised by the service provider. To that end, we believe that the Request for Tender (or RFT) provides a useful model for dynamic web processes. RFTs result in a dynamic service interaction

lifecycle for both the service requestor and provider, in particularly creating an environment where responses can be tailored to a specific service requestor's needs. We have depicted where this work is useful with respect to the spectrum of service provision sophistication (Figure 7.1) in Figure 7.9.

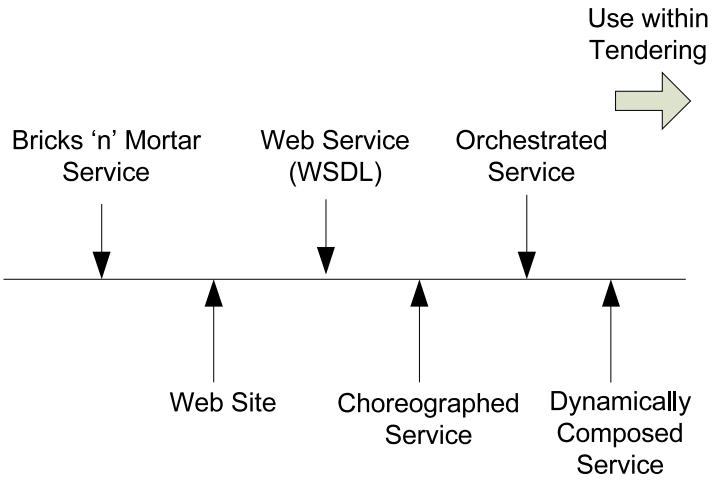


Figure 7.9: Using NFPs in Tendering

The process of tendering is currently used internationally by both businesses (large and small) and government in the procurement of goods and services. It is our assertion that this process (sometimes referred to as a Request for Tender or RFT) is a useful analogy for the service lifecycle. An overview of the tendering process is presented in [95]. We believe that the commonly referenced publish-find-bind model [45] of services is not representative of how business is conducted. Under the publish-find-bind model a service provider publishes the description of their service, requestors use a catalogue to find the service description(s) and then bind to the service that they feel is most appropriate. In our opinion the service provider should not be required to publish a "one size fits all" description into a catalogue.

In contrast, the tendering process provides an opportunity for the prospective purchaser to state its requirements, assess potential providers through the receipt of responses, evaluate responses by subjecting them to specific criteria, and award a contract to the winning tender. A variation on this process is referred to as a "Request for Expressions of Interest" (REOI). This normally involves the publishing of an REOI document. A short list of potential tenderers is selected from the responses to the REOI. These parties are subsequently invited to provide more formal offers. REOI also provides an opportunity for a service requestor to determine the level of interest that exists with respect to their stated needs.

In section 7.5 we provide an insight into our view of the tendering process as an analogy for the service lifecycle. Next, in section 7.6, we present a discussion of how the non-functional properties of services could enhance the tendering process. We have previously highlighted the need for a comprehensive taxonomy for the non-functional properties of services [68]. We believe that such as taxonomy is presented

within this thesis.

7.5 Tendering analogy

As previously stated, we believe that tendering acts as a useful analogy for the service lifecycle (i.e. service discovery, negotiation and invocation). We use Figure 7.10 to present an overview of the tendering process. In this figure we have denoted four major steps within the process: (1) the publishing of the RFT, (2) discovery of the RFT by potential providers, (3) submission of responses, and (4) awarding of a contract. We have purposely not depicted the evaluation of responses to the RFT.

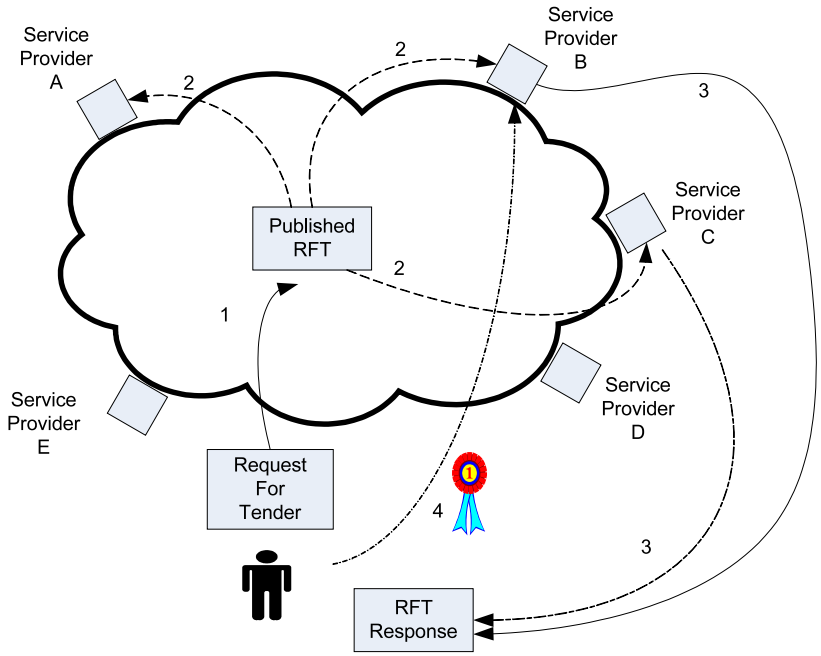


Figure 7.10: The tender process

Automation of the tender process (sometimes referred to as "e-tendering") currently facilitates the publishing and retrieval of tender documents. It doesn't provide a format that enables RFTs or tender responses to be created dynamically through the use of a common vocabulary, or to be reasoned about in some way. Some interesting tender sites in Australia are TenderLink [98] and AusTender [39].

We do not consider that the existing approach of a service provider publishing a detailed service description to a catalogue as being sufficiently flexible for the various service requestor's needs. Service descriptions (or advertisements) attempt to become a "one size fits all" description of the service(s) offered by the provider. Our preference is for the service requestor to initiate the discovery process (through publishing the RFT) and for service providers to respond to each RFT of interest. This approach is only achievable when both the functional and non-functional requirements for the service can be specified by the service requestor, and the response can

be confirmed or further refined, either functionally or non-functionally. One criticism is that this approach results in a state where the service requestor is unaware if a RFT response is being prepared by a service provider(s). The shortened period of initial responses to a REOI would provide a solution to this criticism. We advocate the use of a language that supports such a level of description.

Why take this approach? The primary advantage is that unlike current discovery mechanisms the service descriptions of a provider (i.e. the party that responds to the RFT) are both tailored to the specific request, and published in response to a request (thus avoiding the issue of out-of-date service descriptions). This in turn removes the necessity for a catalogue, it just requires a well-known space for advertising RFTs. This approach would appear to be amenable to the notion of Triple Space Computing [17]. Service providers would also be capable of undertaking dynamic service compositions to meet the requirements stated in tender documents. The tender process also provides a level of transparency for all the service providers. They are usually then aware of all the information that is pertinent to the tender.

As discussed, we believe that the analogy of the RFT provides a useful model for dynamic web processes. It is a slight departure from the existing publish-find-bind model, as the requestor does the publishing, with the provider undertaking the discovery (or finding). With a view to using this type of model with dynamic web processes, we have investigated the content of the interactions that occur between parties under this model. RFTs are rich in non-functional properties. These properties assist service providers with discovery of appropriate RFTs, in guiding how to respond to an RFT, and with providing details within the response to the RFT.

7.6 Non-functional properties in tendering

After having reviewed a significant number of tender documents we believe that there is a clear distinction between the non-functional properties available in tender documents. It would appear that they can be categorised into two areas:

- Those non-functional properties that that relate to the conditions of tendering (i.e. the temporal and locative requirements for submitting a response, laws applicable to the RFT, late response conditions and procedures, and the publisher of the RFT).
- Those non-functional properties that relate to the service delivery outlined within the tender (or an associated appendix). This refers to items such as the location of provision, length of provision, agreed response times, and the warranty period.

Numerous non-functional properties are provided within RFT documents that relate to the conditions of tendering. These include but are not limited to the following:

- Publisher of the RFT, RFT title, and identifier for the RFT.

- The tender lifecycle (dates for issuing the date, closing dates for respondents, selection/evaluation dates, and service commencement).
- The temporal and locative requirements for submitting a response to the RFT (including where and when to perform enquiries with respect to the RFT).
- Definition of terms used within the RFT.
- Pre-conditions to the tender response, selection criteria for response.
- Rights of the party providing the RFT.
- Laws applicable to the RFT (e.g. privacy, auditing, and freedom of information) and applicable jurisdiction(s).
- Retention of intellectual property rights.
- Late response conditions and procedures.
- Tender correction handling procedure.
- Miscellaneous - the bearer of tender response costs, the tender response validity period, general tender submission conditions (including reasons for excluding particular tenders), currency to be used when stating prices, exclusion/inclusion of taxes within prices, consideration for part or joint tenders available, and the language that the response must be provided in.

We are more interested in the non-functional properties that are stated within the requirements section of the RFT. These include but are not limited to the following:

- Location of provision (e.g. Street addresses).
- Ability to perform the services outside normal working hours (e.g. 8am to 4pm).
- Ability to state that you can meet the agreed response time(s). This would need to be stated as specific temporal durations, instants or intervals.
- Length of provision/term of the contract - Examples include the ability to state periods such as 3 years, or alternatively concepts such as 3 years plus two one year options. Provision may also be stated in the form of a temporal interval (e.g. 7:30am to 6:30pm seven days a week including public and other holidays).
- Length of period that a price is valid for - Normally a temporal interval or a temporal duration.
- Discounts that apply to various methods of payment - For example, specific instrument types that receive a discount, or alternatively an early payment discount.

- Warranty periods.
- Term of the contract (e.g. 3 years).
- Related business (e.g. Business "X" is a subsidiary of business "Y").

It is the introduction of a taxonomy for these latter non-functional properties (i.e. those relating to the functional requirements) that will enable the tendering process to be further automated. It does so by enabling the issuer of the RFT to state their functional requirements in conjunction with their non-functional requirements. These descriptions can then be reasoned over by the service provider who formulates a response to the RFT. A requestor could be considered to provide a partial instantiation of the XML Schema presented in Appendix 8.4. This is advertised to some well-known location. Responses can then be a refinement of the originally instantiated models.

The current approach to service discovery involves service requestors using catalogues to determine service providers who are capable of meeting their requirements. We believe that a better approach is to model the service lifecycle on RFTs. This enables service requestors to receive responses that are specific to their requirements. It subsequently results in a catalogue free service environment. Dynamic web processes should be dynamic for all parties concerned. We consider the use of RFTs as a slight departure from the existing publish-find-bind model of service interaction. The approach is advantageous as it is currently used by both business and government, and more importantly, is specifically targeted to the needs of the requestor.

Summary

In this chapter we have attempted to substantiate our work in a number of ways. Firstly, we offered the non-functional properties as a concrete representation within a web service interaction, and subsequently as a semantic annotation to a web service description. Secondly, we provided a discussion of our non-functional properties in electronic tendering. To support this we outlined the conversion process involved in going from everyday service descriptions to XML Schema representations of our ORM models. We believe that these contexts provide a brief outline of the breadth of the applicability of our research.
There are other usages that should also be highlighted. These include:

- YAWL - As previously mentioned, YAWL supports the use of XML Schema types in the definition of input and output tasks within a workflow.
- WS Policy representation of the model. As a minimal number of policy engines are currently available we discounted presenting this context. It should be noted that the concrete representation of our model in XML Schema would easily adapt to the WS Policy standard.

- Business Reporting Language supports the use of XLink (XML Linking Language). Reports could refer to resources (such as non-functional properties) using this standard.
- XQuery - We believe that it is possible to use XQuery over instances of the XML Schema to query the model.

Chapter 8

Epilogue

This thesis has addressed the question of what would be a suitable taxonomy for capturing the non-functional properties of services. The taxonomy that is presented is capable of representing the non-functional properties of services for conventional, electronic and web services in a domain independent manner. We have categorised the non-functional properties according to availability (both temporal and locative), payment, price, discounts, obligations, rights, penalties, trust, security, and quality.

Whilst we acknowledge the importance of service functionality, this thesis is primarily concerned with the non-functional properties of services. A service is *not* a function. It is a function performed on your behalf at a cost. And the cost is not just some monetary price; it is a whole collection of limitations. This thesis is all about these. We believe that a service description is only complete once the non-functional aspects are also expressed.

But why is this needed? Service requestors currently undertake service discovery using proprietary service catalogues, be they physical YellowPages books or web sites. The level of detail provided by those catalogues is insufficient to undertake a form of (semi-)automated reasoning. Whilst a service requestor can ask a service provider for these details in a conventional service context (e.g. by phoning the service provider), electronic representations for services currently lack the descriptive depth to answer many typical questions.

We postulate that the lack of descriptive depth for electronic service descriptions is a result of not having basic concepts for describing services, the heterogeneous nature of services, the inherent complexity of services and that existing approaches to service description do not give appropriate consideration to the context that a requestor brings to the service discovery process.

Our concerns For those existing web service description techniques we have outlined two concerns. Firstly, that conventional services are being ignored for a purely web services view of service description. We referred to this as "web service tunnel vision". Secondly, that these service description techniques are not exploiting the semantic richness of the non-functional properties of services. This we referred to as "semantic myopia". Why are web services so different from conventional services?

We must be reminded that web services will act as an electronic request mechanism for conventional services. We believe that there are numerous benefits for removing the "tunnel vision" of service descriptions to include both web and conventional services.

A service description language that is semantically rich is not without implications. We outlined the increased complexity of user interfaces for discovering services, the additional effort required of the service provider to describe their service, and the increased difficultly in comparing services if the language were flexible. Choosing to ignore both the rich history of conventional services, and the non-functional properties of services (perhaps through deferring to domain specific ontologies, or by a continued functional focus) will result in failure. By implication we believe that accurate service representation promises to reduce the gap between conventional, electronic and web services.

8.1 Criteria for the solution

In Chapter 1 we presented a number of criteria for the solution. The following is a mapping between each criterion and how it was realised within the thesis.

- **Criterion #1: The service description language must support decision-making with respect to services.** *To support any level of decision-making we believe that the service descriptions should have a* formal *grounding. An appropriately chosen formal language has the additional benefits of removing any associated ambiguity, inconsistency and incompleteness. Formalisation also provides an opportunity for reasoning (i.e. the deriving of conclusions) and validation. Utilisation of a formal foundation reduces the complexity associated with implementation, increasing automated support.*

  Our support for decision-making is primarily provided through the use of ORM as a formal grounding, thereby removing any associated ambiguity, inconsistency and incompleteness. Any ORM model can be mapped (using **Rmap**) from its conceptual form to a relational database structure, and corresponding SQL queries may be generated [48].

- **Criterion #2: The service description language must be flexible enough to support the description needs of service providers.** *The service representation language should have adequate* expressive *power to represent all non-functional service properties. Expressiveness allows us to model non-functional properties from many services across various domains. An expressive solution should allow service providers to describe their services in a manner that is easy. Expressiveness as a criterion requires a balance. On one hand we seek to provide the ability to describe the non-functional properties of services in a number of ways (e.g. in describing temporal intervals) versus the ease of discovering services based on non-functional properties that have been expressed differently. The expressiveness of the language must capture*

the static and/or dynamic aspects of the service. The language must also be capable of expressing the relationships between service properties.

Expressiveness has been supported in numerous models within this thesis. For example, service providers are able to use multiple types of temporal intervals - anchored and recurring. In addition, our language captures relationships between service properties. Parts of the model offer recursion, allowing references to other services and their vast set of non-functional properties. This increases the ability to refine service provider candidates during discovery.

- **Criterion #3: The service description language must support the (semi-)automated interactions between service catalogues, service providers and service requestors.** *To achieve this the service representation language should be* machine-processible*. This criterion seeks to reduce the human involvement in the service discovery and comparison tasks. Demonstrating this criterion is a precursor to showing the comprehensiveness of the language. To achieve a level of machine-processing the service description language must be available in a concrete form. In this manner it can be used in interactions between requestors, providers, catalogues and brokers.*

  To achieve this criterion we translated our conceptual models into the form of XML Schema and OWL. It is also available in concrete form within the Web Services Modeling Language (DERI) [102] syntax. By providing these conceptual models in these numerous concrete forms we are able to undertake machine-processing with the language using various techniques.

- **Criterion #4: The service description language must be able to be understood by human beings. In addition, the language should aid the description of the service.** *The service representation language should be* comprehensible*. Human beings should be able to understand the language. The ability for human beings to understand the language may be useful in the context of visual tools that are used for composing, selecting and assembling services. Services that are assembled and provisioned electronically are catered for in the previous criterion. Comprehensibility should aid the service provider in developing a description of their service, reducing the time required to specify its description.*

  We believe that by using names that were extracted during our analysis of numerous existing services that we have maintained a level of comprehensibility. We believe that this will aid both the service provider during description and the service requestor during discovery.

- **Criterion #5: The service description language will not be tied to a particular implementation technology.** *The service description language*

should be conceptual. *This criterion introduces a separation of concerns between the concepts in the language and its implementation. A conceptual language will seek to minimise the number of irrelevant aspects. A solution that is conceptual will not attempt to dictate the implementation.*

We believe that by using ORM we have achieved a level of separation between the language and its implementation (which we show in Chapter 7 in XML Schema and OWL). ORM is a conceptual language and there are numerous implementation technologies that are available.

- **Criterion #6: The service description language must capture and be capable of capturing all relevant service related concepts.** *The service description language should be* suitable. *Suitability ensures that only relevant domain concepts are mapped into the service representation language. Conversely, the language must also be capable of providing all the concepts of a domain.*

  In section 1.4 we offer a series of questions that we use to ensure the suitability of the non-functional properties. These questions were used throughout our analysis of services to filter out non-functional properties that were not present in multiple services across numerous domains.

- **Criterion #7: The service description language must not inhibit extension to itself by the service catalogue or the service provider. This includes inhibiting extensions into domain specifics.** *The service description language should be* extensible. *Extensibility will ensure that domain specific non-functional properties will be able to be included within the language. This criterion also applies to the ability for the non-functional properties of services to evolve over time.*

  We believe the use of the ORM conceptual modelling notation, and careful creation of subtypes now provides an opportunity for extension of our models. We believe that this extension for specific domains would further strengthen our own domain independent research.

8.2 Substantiation

To substantiate our work we presented a discussion surrounding the conversion of our ORM models into XML Schema. This transition provides a machine-processible form for our work. We then took the complex types from the XML Schema and showed their use within Web services, in particular within WSDL. Taking this a step further, we used an automated tool to generate an OWL version of the XML Schema. We subsequently used this OWL version of our work to annotate WSDL according to WSDL-S. Finally, we discussed the use of our non-functional properties

within the context of the tendering process. The transition of the ORM models to XML Schema shows one implementation level view of our work. Future technologies are possible implementation candidates due to the conceptual nature of the work.

8.3 Impact of the research

The primary impact of this research is the inclusion of the work within the non-functional deliverable of the Web Services Modeling Ontology (WSMO) [102], a semantic web initiative developed by the Digital Enterprise Research Institute (DERI). WSMO is one of two major semantic web initiatives, along with OWL-S. DERI's aim is to promote the use of semantics as a fundamental component within current computing. The WSMO aproach provides for the attachment of non-functional properties to four key elements - ontologies, goals, Web services and mediators. DERI's original attempts to describe non-functional properties included the Dublin Core [114] metadata element set (which are all optional), plus a "version" property. Other non-functional property categories (e.g. financial) were outilned but the details were never expanded upon.

In making our work concrete, through the conversion to XML Schema, we were able to further convert our work to the Web Ontology Language (OWL). This conversion to OWL allowed DERI researchers to more easily convert our non-functional properties into the Web Services Modeling Language (WSML). More details about WSML are available in de Bruijn [16].

WSMO's categorisation of non-functional properties is now closely aligned with our work - locative, temporal, availability, obligation, price, payment, discounts, rights, trust, quality of service, security, intellectual property, rewards, provider, measures and currency aspects. A distinct ontology of non-functional properties has now been created in WSML for each of these categories. A full outline of the converted non-functional properties in WSML is available in [102].

We believe that this research has provided a key differentiator for the WSMO standard over other semantic web initatives such as OWL-S. We believe that the inclusion of our work within this research acts as a validation and subsequently we expect to see numerous WSMO semantic web applications making use of these non-functional properties.

8.4 Future work

As with any large piece of research there is possible directions of further work. We believe that these options can be categorised as either research or commercialisation focused. We believe possible future research in this area could include:

- Exploring the impact of this vast number of non-functional properties on the user interface presented to service requestors during service discovery.

- Exploring the areas that we consider to be boundaries to our work (presented

in section 1.5). This includes items such as conditions, procedures and statements.

- Exploring a specific domain's non-functional properties with a view to the impact(s), if any, that is has on this work.
- Exploring the notion of priorities offered by Rosa et al [83]. This prioritisation could provide the ability for a service requestor to denote non-functional properties that are considered more important than others in the service discovery process.
- Exploring the use of ORM as an ontological language.

Possible commercialisation opportunities include:

- Utilisation of the service language presented in this thesis to develop a service comparison tool. Initially, involving a user interface this tool could be used to explore the decision-making of service requestors. An understanding of service requestor decision-making supports future work in the area of automated reasoning.
- Exploring the use of this language within a service catalogue, particularly with a view to providing a user interface that assists service requestors with expression of complex search criteria, thereby increasing the validity of candidate service providers.

Summary

We believe that non-functional properties are an essential component of the characterisation of any service. To support this claim we undertook a significant analysis of services from numerous domains. In the process we have extracted hundreds of non-functional properties that have been presented within this thesis as conceptual models.

We believe that the primary contribution of this research is a domain independent taxonomy capable of representing the non-functional properties of services for conventional, electronic and web services. The concrete representation of this taxonomy (presented in Chapter 7) provides the ability to communicate non-functional properties of services as part of a service description, thereby increasing the efficiency of the service discovery processes. The reasoning that can then be performed over service descriptions reduces the candidate set of services discovered by a requestor. This reduction of candidate services acts to streamline the discovery process. Reasoning can also be performed as part of service comparison and substitution. More generally, the research results in a better understanding of the complex nature of services.

In Chapter 1 we argued that service requestors should be afforded the benefit of rich service descriptions in a similar manner as prospective buyers have access to

product labels. We believe the work contained within this thesis provides service requestors with as much, if not more, information surrounding the non-functional properties relating to the service provider and the service.

Appendix A

Object role modelling notation

We have relied upon the use of Object-Role Modelling (ORM) as a mechanism for portraying the taxonomy that we present herein. ORM is a fact-oriented modelling technique that makes no use of attributes. All facts are represented in the form of entities playing roles. A detailed discussion of ORM is presented in [48]. Looking at Figure 1 we can see ORM entities are represented as named ellipses (e.g. Service, Provider). Attached to the entities are roles that provide a description of the part that an entity plays in a relationship. These relationships, or associations, are represented using one or more role boxes. These are referred to as unary, binary, ternary relationships depending on the number of role boxes.

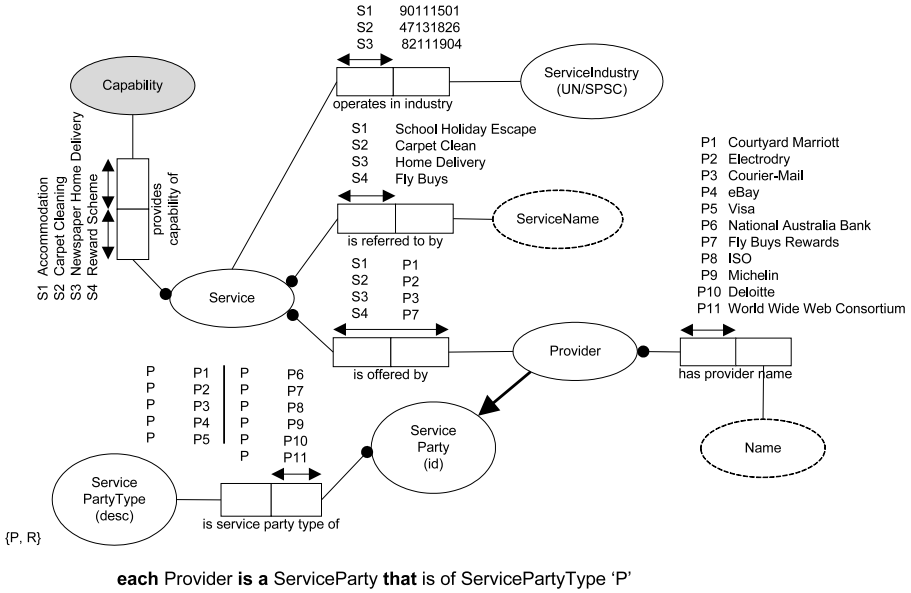


Figure 1: Service provider

Roles are attached to Entity or Value objects using solid lines. Where they appear with a black dot at the point of connection between the line and the entity/value object, then these roles are considered mandatory. Shaded ellipses refer to entities that are defined elsewhere (in another model). We have attempted to provide a partial population of the models as an aid to their understanding. We also make use of ConQuer, a conceptual query language, within this paper as a

means of providing examples of conceptual queries that could be applied to the ORM schemas. A discussion of ConQuer is presented in [12].

Appendix B

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        targetNamespace="http://www.service-description.com/NFP/"
        xmlns:nfp="http://www.service-description.com/NFP/"
        xmlns:iso639-2="http://lcweb.loc.gov/standards/iso639-2/"
        elementFormDefault="qualified" attributeFormDefault="unqualified">

   <xsd:import namespace="http://lcweb.loc.gov/standards/iso639-2/"
schemaLocation="iso639-2.xsd"/>
   <xsd:import namespace="http://www.itu.int/tML/tML-ISO-4217"
schemaLocation="iso4217.xsd"/>
   <xsd:import namespace="http://www.service-description.com/NFP/UNSPSC"
   schemaLocation="unspsc.xsd"/>

   <xsd:complexType name="RequestLocativeAvailability">
       <xsd:sequence>
          <xsd:element name="hasRequestTypeOf" type="nfp:S_RequestType"
maxOccurs="1" minOccurs="1"/>
          <xsd:element name="at" type="nfp:LocativeEntity" maxOccurs="1"
minOccurs="1"/>
       </xsd:sequence>
   </xsd:complexType>

   <xsd:complexType name="A_RequestAvailability">
     <xsd:sequence>
         <xsd:element name="hasRequestLocativeAvailabilityOf"
type="nfp:RequestLocativeAvailability" maxOccurs="1" minOccurs="1"/>
         <xsd:element name="hasTemporalAvailability" maxOccurs="1"
minOccurs="1">
             <xsd:complexType>
                <xsd:choice>
                    <xsd:element name="isContinuouslyAvailableForRequests"
type="xsd:boolean" maxOccurs="1" minOccurs="1"/>
                       <xsd:element name="canOccurAtOrDuring"
type="nfp:TemporalEntity" maxOccurs="unbounded" minOccurs="1"/>
                    <xsd:element
name="hasNegotiableTemporalRequestAvailability" type="xsd:boolean"
maxOccurs="1" minOccurs="1"/>
                </xsd:choice>
            </xsd:complexType>
         </xsd:element>
         <xsd:element name="canBeLimitedTo" type="nfp:LocativeEntity"
```

```
maxOccurs="unbounded" minOccurs="0"/>
</xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="ProvisionLocativeAvailability">
    <xsd:sequence>
      <xsd:element name="hasProvisionLocation">
        <xsd:complexType>
          <xsd:choice>
            <xsd:element name="canBeProvidedTo" type="nfp:LocativeEntity"
maxOccurs="1" minOccurs="1"/>
            <xsd:element name="isTheTypeOfLocationTheServiceCanBeProvidedTo"
type="nfp:L_LocativeEntityType" maxOccurs="1" minOccurs="1"/>
          </xsd:choice>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="A_ProvisionAvailability">
    <xsd:sequence>
      <xsd:element name="hasProvisionLocativeAvailabilityOf"
type="nfp:ProvisionLocativeAvailability" maxOccurs="1" minOccurs="1"/>
      <xsd:element name="hasTemporalAvailability" maxOccurs="1" minOccurs="1">
        <xsd:complexType>
          <xsd:choice>
            <xsd:element name="isContinuouslyAvailableForProvision"
type="xsd:boolean" maxOccurs="1" minOccurs="1"/>
            <xsd:element name="canOccurAtOrDuring" type="nfp:TemporalEntity"
maxOccurs="unbounded" minOccurs="1"/>
            <xsd:element name="hasNegotiableTemporalProvisionAvailability"
type="xsd:boolean" maxOccurs="1" minOccurs="1"/>
          </xsd:choice>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="Availability">
      <xsd:sequence>
          <xsd:element name="isAvailableAt" type="nfp:LocativeEntity"
maxOccurs="1" minOccurs="1"/>
          <xsd:element name="during" type="nfp:TemporalEntity" maxOccurs="1"
minOccurs="1"/>
      </xsd:sequence>
  </xsd:complexType>

    <xsd:complexType name="Capability">
        <xsd:simpleContent>
            <xsd:extension base="xsd:anyURI"/>
        </xsd:simpleContent>
    </xsd:complexType>

    <xsd:complexType name="Condition">
```

```
        <xsd:simpleContent>
            <xsd:extension base="xsd:anyURI"/>
        </xsd:simpleContent>
    </xsd:complexType>

    <xsd:complexType name="Statement">
        <xsd:simpleContent>
            <xsd:extension base="xsd:anyURI"/>
        </xsd:simpleContent>
    </xsd:complexType>

    <xsd:complexType name="Penalty">
        <xsd:sequence>
         <xsd:element name="hasPenaltyOf">
             <xsd:complexType>
                 <xsd:choice>
                     <xsd:element name="FinancialPenalty"
type="nfp:P_FinancialPenalty"/>
                     <xsd:element name="InvoluntarySuspensionPenalty"
type="nfp:P_InvoluntarySuspensionPenalty"/>
                     <xsd:element name="TerminationPenalty"
type="nfp:P_TerminationPenalty"/>
                     <xsd:element name="LossOfRightPenalty"
type="nfp:P_LossOfRightPenalty"/>
                   </xsd:choice>
               </xsd:complexType>
          </xsd:element>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="P_Penalty">
        <xsd:sequence>
           <xsd:element name="isImposedForNonComplianceTo"
type="nfp:Obligation" maxOccurs="unbounded" minOccurs="1"/>
         <xsd:element name="resultsInAdditionalLossOf" type="nfp:Right"
maxOccurs="unbounded" minOccurs="0"/>
         <xsd:element name="hasPenaltyConditionOf" type="nfp:Condition"
maxOccurs="unbounded" minOccurs="0"/>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="P_FinancialPenalty">
        <xsd:complexContent>
            <xsd:extension base="nfp:P_Penalty">
                <xsd:sequence>
                    <xsd:element name="hasFinancialPenaltyOf"
type="nfp:PricingObligation" maxOccurs="1" minOccurs="1"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="P_InvoluntarySuspensionPenalty">
        <xsd:complexContent>
```

```
            <xsd:extension base="nfp:P_Penalty">
                <xsd:sequence>
                    <xsd:element name="invokesServiceProviderSuspensionOf"
type="nfp:RightOfSuspension" maxOccurs="1" minOccurs="1"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="P_TerminationPenalty">
        <xsd:complexContent>
            <xsd:extension base="nfp:P_Penalty">
                <xsd:sequence>
                    <xsd:element name="invokesServiceProviderTerminationOf"
type="nfp:RightOfTermination" maxOccurs="1" minOccurs="1"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="P_LossOfRightPenalty">
        <xsd:complexContent>
            <xsd:extension base="nfp:P_Penalty">
                <xsd:sequence>
                    <xsd:element name="resultsInLossOf" type="nfp:Right"
maxOccurs="unbounded" minOccurs="1"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="Discount">
        <xsd:sequence>
         <xsd:element name="hasDiscountOf">
             <xsd:complexType>
                 <xsd:choice>
                     <xsd:element name="PayeeDiscount"
type="nfp:D_PayeeDiscount"/>
                     <xsd:element name="PaymentDiscount"
type="nfp:D_PaymentDiscount"/>
                 </xsd:choice>
               </xsd:complexType>
          </xsd:element>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="D_Discount">
        <xsd:sequence>
           <xsd:element name="hasDiscountConditionOf" type="nfp:Condition"
maxOccurs="unbounded" minOccurs="0"/>
         <xsd:element name="hasDiscountPriceOf" type="nfp:D_DiscountAmount"
maxOccurs="unbounded" minOccurs="0"/>
         <xsd:element name="hasResultingDiscountPriceOf"
type="nfp:D_ResultingDiscountedPrice" maxOccurs="unbounded" minOccurs="0"/>
```

```
         <xsd:element name="hasDiscountAvailabilityOf"
type="nfp:Availability" maxOccurs="unbounded" minOccurs="1"/>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="D_PayeeDiscount">
        <xsd:complexContent>
            <xsd:extension base="nfp:D_Discount">
                <xsd:sequence>
            <xsd:element name="hasPayeeDiscountOf">
                 <xsd:complexType>
                     <xsd:choice>
                         <xsd:element name="StudentDiscount"
type="nfp:D_StudentDiscount"/>
                         <xsd:element name="MembershipDiscount"
type="nfp:D_MembershipDiscount"/>
                        <xsd:element name="ShareholderDiscount"
type="nfp:D_ShareholderDiscount"/>
                         <xsd:element name="AgeGroupDiscount"
type="nfp:D_AgeGroupDiscount"/>
                     </xsd:choice>
                </xsd:complexType>
             </xsd:element>
          </xsd:sequence>
</xsd:extension>
  </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="D_StudentDiscount">
        <xsd:complexContent>
            <xsd:extension base="nfp:D_PayeeDiscount">
                <xsd:sequence>
                    <xsd:element name="applicableToSchoolStudents"
type="xsd:boolean" maxOccurs="1" minOccurs="1"/>
                    <xsd:element name="applicableToFulltimeUniversityStudents"
type="xsd:boolean" maxOccurs="1" minOccurs="1"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="D_MembershipDiscount">
        <xsd:complexContent>
            <xsd:extension base="nfp:D_PayeeDiscount">
                <xsd:sequence>
                    <xsd:element name="isAvailableToHoldersOf"
type="nfp:O_Membership" maxOccurs="1" minOccurs="1"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="D_ShareholderDiscount">
        <xsd:complexContent>
```

```
            <xsd:extension base="nfp:D_PayeeDiscount">
                <xsd:sequence>
                    <xsd:element name="availableToShareholdersOf"
type="nfp:P_Provider" maxOccurs="1" minOccurs="1"/>
                    <xsd:element
name="availableToShareholdersWithMinimumNumberOfUnitsOf"
type="xsd:nonNegativeInteger" maxOccurs="1" minOccurs="1"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="D_AgeGroupDiscount">
        <xsd:complexContent>
            <xsd:extension base="nfp:D_PayeeDiscount">
                <xsd:sequence>
                    <xsd:element name="hasAgeGroupNameOf" type="xsd:string"
maxOccurs="1" minOccurs="0"/>
                    <xsd:element name="hasAgeGroupFromValueOf"
type="xsd:nonNegativeInteger" maxOccurs="1" minOccurs="1"/>
                    <xsd:element name="hasAgeGroupToValueOf"
type="xsd:nonNegativeInteger" maxOccurs="1" minOccurs="1"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="D_DiscountAmount">
        <xsd:sequence>
            <xsd:element name="hasDiscountAmountOf" type="nfp:T_Price"
maxOccurs="1" minOccurs="1"/>
            <xsd:element name="forService" type="nfp:S_Service" maxOccurs="1"
minOccurs="1"/>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="D_ResultingDiscountedPrice">
        <xsd:sequence>
            <xsd:element name="resultsInDiscountedPriceOf" type="nfp:T_Price"
maxOccurs="1" minOccurs="1"/>
            <xsd:element name="forService" type="nfp:S_Service" maxOccurs="1"
minOccurs="1"/>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="D_PaymentDiscount">
        <xsd:complexContent>
            <xsd:extension base="nfp:D_Discount">
                <xsd:sequence>
            <xsd:element name="hasPaymentDiscountOf">
                 <xsd:complexType>
                     <xsd:choice>
                        <xsd:element name="PaymentInstrumentTypeDiscount"
type="nfp:D_PaymentInstrumentTypeDiscount"/>
```

```
                        <xsd:element name="PaymentLocationTypeDiscount"
type="nfp:D_PaymentLocationTypeDiscount"/>
                        <xsd:element name="CouponPaymentDiscount"
type="nfp:D_CouponPaymentDiscount"/>
                        <xsd:element name="EarlyPaymentDiscount"
type="nfp:D_EarlyPaymentDiscount"/>
<xsd:element name="VolumeInvocationTypeDiscount"
type="nfp:D_VolumeInvocationTypeDiscount"/>
                    </xsd:choice>
                 </xsd:complexType>
             </xsd:element>
            <xsd:element name="hasMinimumPriceRequiredToReceiveDiscountOf"
type="nfp:T_Price" maxOccurs="1" minOccurs="0"/>
        </xsd:sequence>
</xsd:extension>
</xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="D_PaymentInstrumentTypeDiscount">
        <xsd:complexContent>
            <xsd:extension base="nfp:D_PaymentDiscount">
                <xsd:sequence>
                    <xsd:element name="offersPaymentInstrumentTypeDiscountOf"
type="nfp:PaymentInstrumentType" maxOccurs="1" minOccurs="1"/>

                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="D_PaymentLocationTypeDiscount">
        <xsd:complexContent>
            <xsd:extension base="nfp:D_PaymentDiscount">
                <xsd:sequence>
                    <xsd:element name="offersPaymentLocationTypeDiscountFor"
type="nfp:L_LocativeEntityType" maxOccurs="1" minOccurs="0"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="D_CouponPaymentDiscount">
        <xsd:complexContent>
            <xsd:extension base="nfp:D_PaymentDiscount">
                <xsd:sequence>
                    <xsd:element name="hasValidityPeriodOf"
type="nfp:TemporalEntity" maxOccurs="1" minOccurs="1"/>
                    <xsd:element name="isIssuedBy" type="nfp:P_Provider"
maxOccurs="1" minOccurs="1"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
```

```
    <xsd:complexType name="D_EarlyPaymentDiscount">
        <xsd:complexContent>
            <xsd:extension base="nfp:D_PaymentDiscount">
                <xsd:sequence>
                    <xsd:element name="hasEarlyPaymentDiscountOf">
                        <xsd:complexType>
                            <xsd:choice>
                                <xsd:element name="hasEarlyPaymentOffsetOf"
type="nfp:T_TemporalDuration"/>
                                <xsd:element name="hasEarlyPaymentScheduleOf"
type="nfp:PaymentSchedule"/>
                            </xsd:choice>
                        </xsd:complexType>
                    </xsd:element>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="D_VolumeInvocationTypeDiscount">
        <xsd:complexContent>
            <xsd:extension base="nfp:D_PaymentDiscount">
                <xsd:sequence>
                    <xsd:element name="hasVolumeInvocationRangeOf"
type="nfp:D_VolumeInvocationRange" maxOccurs="unbounded" minOccurs="2"/>
                    <xsd:element name="hasPriceReceivedForInvocationsOf"
type="nfp:P_Price"  maxOccurs="unbounded" minOccurs="1"/>
                    <xsd:element name="hasInvocationTemporalValidityOf"
type="nfp:T_TemporalEntity"  maxOccurs="unbounded" minOccurs="0"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="D_VolumeInvocationRange">
        <xsd:sequence>
            <xsd:element name="hasInvocationBoundaryOf"
type="nfp:D_InvocationBoundaryPosition" maxOccurs="1" minOccurs="1"/>
            <xsd:element name="forPositionOf" type="xsd:nonNegativeInteger"
maxOccurs="1" minOccurs="1"/>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:simpleType name="D_InvocationBoundaryPosition">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="start"/>
            <xsd:enumeration value="end"/>
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:complexType name="RegisteredIPRight">
     <xsd:sequence>
         <xsd:element name="hasIPRightOf">
             <xsd:complexType>
```

```
                <xsd:choice>
                    <xsd:element name="TrademarkOrDesign"
type="nfp:TrademarkOrDesign"/>
<xsd:element name="Patent" type="nfp:Patent"/>
</xsd:choice>
  </xsd:complexType>
    </xsd:element>
    <xsd:element name="hasApplicationNumberOf" type="xsd:decimal"
maxOccurs="1" minOccurs="1"/>
    <xsd:element name="hasRightStatusOf" type="nfp:IPRightStatus"
maxOccurs="1" minOccurs="1"/>
    <xsd:element name="hasRegionOfProtectionOf" type="nfp:L_Region"
maxOccurs="1" minOccurs="0"/>
    <xsd:element name="hasCountryOfOriginOf" type="nfp:L_Region"
maxOccurs="1" minOccurs="0"/>
    <xsd:element name="hasAgentAddressOf" type="nfp:L_Address" maxOccurs="1"
minOccurs="0"/>
    <xsd:element name="hasAgentOf" type="nfp:Agent" maxOccurs="1"
minOccurs="0"/>
    <xsd:element name="isDetailsFor" type="nfp:LocativeEntity" maxOccurs="1"
minOccurs="0"/>
    <xsd:element name="hasRegistrationDateOf" type="nfp:T_TemporalDate"
maxOccurs="1" minOccurs="0"/>
    <xsd:element name="hasLodgingDateOf" type="nfp:T_TemporalDate"
maxOccurs="1" minOccurs="1"/>
    <xsd:element name="hasOwnerOf" type="nfp:P_Provider" maxOccurs="1"
minOccurs="1"/>
   </xsd:sequence>
   </xsd:complexType>

<xsd:complexType name="TrademarkOrDesign">
     <xsd:sequence>
         <xsd:element name="hasTrademarkOrDesignOf">
             <xsd:complexType>
                 <xsd:choice>
                     <xsd:element name="Trademark" type="nfp:Trademark"/>
<xsd:element name="Design" type="nfp:Design"/>
</xsd:choice>
  </xsd:complexType>
    </xsd:element>
    <xsd:element name="hasDurationOfProtectionOf"
type="nfp:T_TemporalDuration" maxOccurs="1" minOccurs="1"/>
   </xsd:sequence>
   </xsd:complexType>

<xsd:complexType name="Trademark">
<xsd:sequence>
<xsd:element name="hasWordmarkOf" type="xsd:anyURI" maxOccurs="1"
minOccurs="1"/>
<xsd:element name="hasTrademarkTypeOf" type="nfp:TrademarkType"
maxOccurs="1" minOccurs="1"/>
         <xsd:element name="hasFigurativeMarkClassificationOf"
type="nfp:FigurativeMarkClassification-Vienna" maxOccurs="1" minOccurs="1"/>
<xsd:element name="hasTrademarkClassificationOf">
```

```
<xsd:complexType>
<xsd:choice>
                                    <xsd:element name="hasGoodsClassificationOf"
type="nfp:TrademarkClassification-Nice"/>
<xsd:element name="hasServiceClassificationOf"
type="nfp:TrademarkClassification-Nice"/>
            </xsd:choice>
</xsd:complexType>
</xsd:element>
</xsd:sequence>
</xsd:complexType>

<xsd:complexType name="Design">
<xsd:sequence>
<xsd:element name="hasDesignClassificationOf" type="nfp:IDCLocarno"
maxOccurs="1" minOccurs="1"/>
</xsd:sequence>
</xsd:complexType>

<xsd:complexType name="Patent">
<xsd:sequence>
<xsd:element name="hasPatentClassificationOf"
type="nfp:PatentClassification-IPCStrasbourg" maxOccurs="1" minOccurs="1"/>
<xsd:element name="hasTitleOf" type="xsd:string" maxOccurs="1"
minOccurs="1"/>
<xsd:element name="hasInventorOf" type="xsd:string" maxOccurs="1"
minOccurs="1"/>
</xsd:sequence>
</xsd:complexType>

<xsd:simpleType name="IDCLocarno">
 <xsd:restriction base="xsd:string"/>
</xsd:simpleType>

<xsd:simpleType name="TrademarkClassification-Nice">
 <xsd:restriction base="xsd:string"/>
</xsd:simpleType>

<xsd:simpleType name="FigurativeMarkClassification-Vienna">
 <xsd:restriction base="xsd:string"/>
</xsd:simpleType>

<xsd:simpleType name="PatentClassification-IPCStrasbourg">
 <xsd:restriction base="xsd:string"/>
</xsd:simpleType>

<xsd:simpleType name="IPRightStatus">
 <xsd:restriction base="xsd:string"/>
</xsd:simpleType>

<xsd:simpleType name="Agent">
 <xsd:restriction base="xsd:string"/>
</xsd:simpleType>
```

```
    <xsd:simpleType name="TrademarkType">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="Trademark"/>
            <xsd:enumeration value="ServiceMark"/>
            <xsd:enumeration value="DressMark"/>
            <xsd:enumeration value="CollectiveMark"/>
            <xsd:enumeration value="CertificationMark"/>
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:complexType name="LocativeEntity">
     <xsd:sequence>
         <xsd:element name="hasLocativeValue">
             <xsd:complexType>
                 <xsd:choice>
                   <xsd:element name="Point" type="nfp:L_Point"/>
                   <xsd:element name="MovingPoint" type="nfp:L_MovingPoint"/>

                   <xsd:element name="Route" type="nfp:L_Route"/>
                   <xsd:element name="Region" type="nfp:L_Region"/>
                   <xsd:element name="Address" type="nfp:L_Address"/>
                   <xsd:element name="StreetAddress"
type="nfp:L_StreetAddress"/>
                   <xsd:element name="PostboxAddress"
type="nfp:L_PostboxAddress"/>
                         <xsd:element name="URI" type="nfp:L_URI"/>
                         <xsd:element name="PhoneNumber"
type="nfp:L_PhoneNumber"/>
                         <xsd:element name="IPAddress" type="nfp:L_IPAddress"/>
                         <xsd:element name="EthernetAddress"
type="nfp:L_EthernetAddress"/>
                         <xsd:element name="Spectrum" type="nfp:L_Spectrum"/>
                         <xsd:element name="StreetDirectoryReference"
type="nfp:L_StreetDirectoryReference"/>
                    </xsd:choice>
                </xsd:complexType>
            </xsd:element>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="L_LocativeEntity">
        <xsd:sequence>
            <xsd:element name="hasLocativeEntityConditionOf"
type="nfp:Condition"/>
            <xsd:element name="hasLocativeCommonNameOf"
type="nfp:L_LocativeCommon" maxOccurs="unbounded" minOccurs="0"/>
            <xsd:element name="supportsWrittenLanguageOf" type="iso639-
2:CodeType" maxOccurs="unbounded" minOccurs="0"/>
            <xsd:element name="supportsSpokenLanguageOf" type="iso639-
2:CodeType" maxOccurs="unbounded" minOccurs="0"/>
            <xsd:element name="canBeCommunicatedWithAccordingTo"
type="nfp:Q_Standard" maxOccurs="unbounded" minOccurs="0"/>
            <xsd:element name="hasIdentificationRequirement"
type="nfp:IdentificationRequirement" maxOccurs="1" minOccurs="0"/>
```

```
            <xsd:element name="providesConfidentialityOf"
type="nfp:Confidentiality" maxOccurs="1" minOccurs="0"/>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="L_LocativeCommon">
            <xsd:sequence>
                <xsd:element name="hasLocativeCommonNameOf" type="xsd:string"
maxOccurs="1" minOccurs="1"/>
                <xsd:element name="appliesToLocation"
type="nfp:L_LocativeEntity" maxOccurs="1" minOccurs="0"/>
            </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="L_Point">
        <xsd:complexContent>
            <xsd:extension base="nfp:L_LocativeEntity">
                <xsd:sequence>
                    <xsd:element name="hasLatitudeOf" type="nfp:L_Coordinate"
maxOccurs="1" minOccurs="1"/>
                    <xsd:element name="hasLongitudeOf" type="nfp:L_Coordinate"
maxOccurs="1" minOccurs="1"/>
                    <xsd:element name="hasAltitudeOf"
type="xsd:nonNegativeInteger" maxOccurs="1" minOccurs="0"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="L_Coordinate">
        <xsd:sequence>
            <xsd:element name="hasDegreesOf" maxOccurs="1" minOccurs="1">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:integer">
                        <xsd:minExclusive value="-180"/>
                        <xsd:maxExclusive value="180"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
            <xsd:element name="hasMinutesOf" type="nfp:T_Minutes"
maxOccurs="1" minOccurs="1"/>
            <xsd:element name="hasSecondsOf" type="nfp:T_Seconds"
maxOccurs="1" minOccurs="1"/>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="L_MovingPoint">
        <xsd:complexContent>
            <xsd:extension base="nfp:L_Point">
                <xsd:sequence>
                    <xsd:element name="hasVelocityOf"
type="xsd:nonNegativeInteger" maxOccurs="1" minOccurs="1"/>
                    <xsd:element name="hasDirectionOf" type="nfp:L_Direction"
maxOccurs="1" minOccurs="1"/>
```

```
                    <xsd:element name="hasTimeOf"
type="nfp:T_AnchoredPointinTime" maxOccurs="1" minOccurs="1"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="L_Route">
        <xsd:complexContent>
            <xsd:extension base="nfp:L_LocativeEntity">
                <xsd:sequence>
                    <xsd:element name="hasIndicativeRouteTypeOf" maxOccurs="1"
minOccurs="1">
                       <xsd:simpleType>
                           <xsd:restriction base="xsd:string">
                               <xsd:enumeration value="Bus"/>
                               <xsd:enumeration value="Plane"/>
                               <xsd:enumeration value="Railway"/>
                               <xsd:enumeration value="Highway"/>
                           </xsd:restriction>
                      </xsd:simpleType>
                    </xsd:element>
                    <xsd:element name="hasRouteNameOf" type="xsd:string"
maxOccurs="1" minOccurs="1"/>
                    <xsd:element name="hasRouteSpecificationOf"
type="nfp:L_RouteSpecification" maxOccurs="unbounded" minOccurs="0"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="L_RouteSpecification">
         <xsd:sequence>
             <xsd:element name="hasSpecificationOf"
type="nfp:L_RouteSpecificationType" maxOccurs="unbounded" minOccurs="2"/>
             <xsd:element name="hasRouteSpecificationOperationOf"
type="nfp:L_RouteSpecificationOperation" maxOccurs="unbounded" minOccurs="0"/>
         </xsd:sequence>
    </xsd:complexType>

     <xsd:complexType name="L_RouteSpecificationOperation">
         <xsd:sequence>
             <xsd:element name="hasOperationTypeOf" type="nfp:L_OperationType"
maxOccurs="1" minOccurs="1"/>
             <xsd:element name="usingPointOf" type="nfp:L_Point" maxOccurs="1"
minOccurs="1"/>
         </xsd:sequence>
    </xsd:complexType>

     <xsd:complexType name="L_RouteSpecificationType">
         <xsd:sequence>
             <xsd:element name="hasPointOf" type="nfp:L_Point" maxOccurs="1"
minOccurs="1"/>
             <xsd:element name="hasOrderOf" type="xsd:nonNegativeInteger"
```

```
maxOccurs="1" minOccurs="1"/>
         </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="L_Region">
        <xsd:complexContent>
            <xsd:extension base="nfp:L_LocativeEntity">
                <xsd:sequence>
                    <xsd:element name="hasRegionTypeOf" maxOccurs="1"
minOccurs="1">
                       <xsd:simpleType>
                           <xsd:restriction base="xsd:string">
                               <xsd:enumeration value="Country"/>
                               <xsd:enumeration value="State"/>
                               <xsd:enumeration value="Territory"/>
                               <xsd:enumeration value="Province"/>
                               <xsd:enumeration value="Suburb"/>
                               <xsd:enumeration value="County"/>
                               <xsd:enumeration value="Republic"/>
                               <xsd:enumeration value="Continent"/>
                           </xsd:restriction>
                      </xsd:simpleType>
                    </xsd:element>
                    <xsd:element name="hasRegionNameOf" type="xsd:string"
maxOccurs="1" minOccurs="1"/>
                    <xsd:element name="hasAliasOf" type="nfp:L_Point"
maxOccurs="1" minOccurs="0"/>
                    <xsd:element name="hasRegionSpecificationOf"
type="nfp:L_RegionSpecification" maxOccurs="unbounded" minOccurs="0"/>

                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="L_RegionSpecification">
         <xsd:sequence>
             <xsd:element name="hasSpecificationOf"
type="nfp:L_RegionSpecificationType" maxOccurs="unbounded" minOccurs="3"/>
<xsd:element name="hasRegionSpecificationOperationOf"
type="nfp:L_RegionSpecificationOperation" maxOccurs="unbounded" minOccurs="0"/>
         </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="L_RegionSpecificationOperation">
         <xsd:sequence>
             <xsd:element name="hasOperationTypeOf" type="nfp:L_OperationType"
maxOccurs="1" minOccurs="1"/>
             <xsd:element name="usingRegionSpecificationOf"
type="nfp:L_RegionSpecification" maxOccurs="1" minOccurs="1"/>
         </xsd:sequence>
    </xsd:complexType>

     <xsd:complexType name="L_RegionSpecificationType">
```

```
        <xsd:sequence>
            <xsd:element name="hasBoundaryPointOf" type="nfp:L_Point"
maxOccurs="1" minOccurs="1"/>
             <xsd:element name="hasOrderOf" type="xsd:nonNegativeInteger"
maxOccurs="1" minOccurs="1"/>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="L_Address">
        <xsd:complexContent>
            <xsd:extension base="nfp:L_LocativeEntity">
                <xsd:sequence>
                    <xsd:element name="hasSuburbOf" type="nfp:L_Region"
maxOccurs="1" minOccurs="0"/>
                    <xsd:element name="hasCountyOf" type="nfp:L_Region"
maxOccurs="1" minOccurs="0"/>
                    <xsd:element name="hasProvinceOf" type="nfp:L_Region"
maxOccurs="1" minOccurs="0"/>
                    <xsd:element name="hasStateOf" type="nfp:L_Region"
maxOccurs="1" minOccurs="0"/>
                    <xsd:element name="hasTerritoryOf" type="nfp:L_Region"
maxOccurs="1" minOccurs="0"/>
                    <xsd:element name="hasCityOf" type="nfp:L_Region"
maxOccurs="1" minOccurs="1"/>
                    <xsd:element name="hasCountryOf" type="nfp:L_Region"
maxOccurs="1" minOccurs="1"/>
                    <xsd:element name="hasPostcodeOf" type="nfp:L_Postcode"
maxOccurs="1" minOccurs="1"/>
                    <xsd:element name="hasAddresseeOf" type="nfp:L_Addressee"
maxOccurs="1" minOccurs="0"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="L_PostboxAddress">
        <xsd:complexContent>
            <xsd:extension base="nfp:L_Address">
                <xsd:sequence>
                    <xsd:element name="hasPostBoxTypeOf">
                        <xsd:simpleType>
                           <xsd:restriction base="xsd:string">
                               <xsd:enumeration value="General"/>
                               <xsd:enumeration value="Private"/>
                               <xsd:enumeration value="LockedBag"/>
                           </xsd:restriction>
                      </xsd:simpleType>
                    </xsd:element>
                    <xsd:element name="hasPostboxNumberOf"
type="xsd:nonNegativeInteger"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
```

```
    <xsd:complexType name="L_StreetAddress">
        <xsd:complexContent>
            <xsd:extension base="nfp:L_Address">
                <xsd:sequence>
                    <xsd:element name="hasStreetType" type="nfp:StreetType"
maxOccurs="1" minOccurs="1"/>
                    <xsd:element name="hasStreetNameOf" type="xsd:string"
maxOccurs="1" minOccurs="1"/>
                    <xsd:element name="hasStreetNumberOf"
type="xsd:nonNegativeInteger"/>
                    <xsd:element name="hasBuildingNumberOf"
type="xsd:nonNegativeInteger"/>
                    <xsd:element name="hasLevelOf"
type="xsd:nonNegativeInteger"/>
                    <xsd:element name="hasUnitNumberOf"
type="xsd:nonNegativeInteger"/>
                    <xsd:element name="hasRoomNumberOf"
type="xsd:nonNegativeInteger"/>
                    <xsd:element name="hasApartmentNumberOf"
type="xsd:nonNegativeInteger"/>
                    <xsd:element name="hasSuiteNumberOf"
type="xsd:nonNegativeInteger"/>
                    <xsd:element name="hasStreetDirectoryReferenceOf"
type="nfp:L_StreetDirectoryReference"/>
                    <xsd:element name="hasProximityOf" type="nfp:L_Proximity"/>
                    <xsd:element name="hasMapReferenceOf" type="xsd:anyURI"/>

                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:simpleType name="StreetType">
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value="Circuit"/>
        <xsd:enumeration value="Court"/>
        <xsd:enumeration value="Crescent"/>
        <xsd:enumeration value="Place"/>
        <xsd:enumeration value="Street"/>
        <xsd:enumeration value="Road"/>
        <xsd:enumeration value="Lane"/>
        <xsd:enumeration value="Way"/>
        <xsd:enumeration value="Avenue"/>
        <xsd:enumeration value="Esplanade"/>
        <xsd:enumeration value="Promenade"/>
        <xsd:enumeration value="Chase"/>
      </xsd:restriction>
    </xsd:simpleType>

    <xsd:simpleType name="L_OperationType">
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value="Refinement"/>
        <xsd:enumeration value="Exception"/>
```

```
      </xsd:restriction>
    </xsd:simpleType>

    <xsd:complexType name="L_Proximity">
         <xsd:sequence>
             <xsd:element name="is" type="nfp:Preposition"/>
             <xsd:element name="toAddress" type="nfp:L_StreetAddress"/>
         </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="L_Addressee">
         <xsd:sequence>
             <xsd:element name="hasDepartmentNameOf" type="xsd:string"
maxOccurs="1" minOccurs="0"/>
             <xsd:element name="hasAddresseeNameOf" type="xsd:string"
maxOccurs="1" minOccurs="0"/>
             <xsd:element name="hasAddresseeFunctionalTitleOf"
type="xsd:string" maxOccurs="1" minOccurs="0"/>
             <xsd:element name="hasAddresseeProfessionalTitleOf"
type="xsd:string" maxOccurs="1" minOccurs="0"/>
             <xsd:element name="hasOrganisationNameOf" type="xsd:string"
maxOccurs="1" minOccurs="0"/>
         </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="L_PhoneNumber">
        <xsd:complexContent>
            <xsd:extension base="nfp:L_LocativeEntity">
                <xsd:sequence>
                    <xsd:element name="PhoneNumberType" maxOccurs="1"
minOccurs="1">
                        <xsd:simpleType>
                           <xsd:restriction base="xsd:string">
                               <xsd:enumeration value="Mobile/Cell"/>
                               <xsd:enumeration value="FixedLine"/>
                           </xsd:restriction>
                      </xsd:simpleType>
                    </xsd:element>
                    <xsd:element name="InteractionType" maxOccurs="1"
minOccurs="1">
                        <xsd:simpleType>
                           <xsd:restriction base="xsd:string">
                               <xsd:enumeration value="Voice"/>
                               <xsd:enumeration value="Modem"/>
                               <xsd:enumeration value="SMSTextMessaging"/>
                               <xsd:enumeration value="Facsimile"/>
                               <xsd:enumeration value="Telex"/>
                           </xsd:restriction>
                      </xsd:simpleType>
                    </xsd:element>
                    <xsd:element name="tollFreeCallForCallersFromRegionOf"
type="nfp:L_Region" maxOccurs="1" minOccurs="0"/>
                    <xsd:element name="hasCountryCodeOf"
type="xsd:nonNegativeInteger" maxOccurs="1" minOccurs="0"/>
```

```
                    <xsd:element name="hasNationalDirectDialPrefixOf"
type="xsd:nonNegativeInteger" maxOccurs="1" minOccurs="0"/>

                    <xsd:element name="hasCityOrAreaCodeOf"
type="xsd:nonNegativeInteger" maxOccurs="1" minOccurs="0"/>

                    <xsd:element name="hasLocalNumberOf"
type="xsd:nonNegativeInteger" maxOccurs="1" minOccurs="1"/>
                    <xsd:element name="hasInternationalPrefixFromRegionOf"
type="nfp:L_InternationalPrefixForRegion" maxOccurs="unbounded" minOccurs="0"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="L_InternationalPrefixForRegion">
         <xsd:sequence>
             <xsd:element name="hasInternationalDirectDialPrefixOf"
type="xsd:nonNegativeInteger" maxOccurs="1" minOccurs="1"/>
             <xsd:element name="forCallersFromRegion" type="nfp:L_Region"
minOccurs="1" maxOccurs="1"/>
         </xsd:sequence>
    </xsd:complexType>

    <xsd:simpleType name="L_URI">
        <xsd:restriction base="xsd:anyURI"/>
    </xsd:simpleType>

    <xsd:complexType name="L_IPAddress">
        <xsd:complexContent>
            <xsd:extension base="nfp:L_LocativeEntity">
                <xsd:sequence>
                    <xsd:element name="hasClassAAddressOf"
type="nfp:L_IPAddressNumber" minOccurs="1" maxOccurs="1"/>
                    <xsd:element name="hasClassBAddressOf"
type="nfp:L_IPAddressNumber" minOccurs="1" maxOccurs="1"/>
                    <xsd:element name="hasClassCAddressOf"
type="nfp:L_IPAddressNumber" minOccurs="1" maxOccurs="1"/>
                    <xsd:element name="hasLocalAddressOf"
type="nfp:L_IPAddressNumber" minOccurs="1" maxOccurs="1"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="L_EthernetAddress">
        <xsd:complexContent>
            <xsd:extension base="nfp:L_LocativeEntity">
                <xsd:sequence>
                    <xsd:element name="isUniversallyAdministered"
type="xsd:boolean" minOccurs="1" maxOccurs="1"/>
                    <xsd:element name="isGroupAddress" type="xsd:boolean"
minOccurs="1" maxOccurs="1"/>
                    <xsd:element name="represents" type="nfp:P_Provider"
```

```
minOccurs="1" maxOccurs="1"/>
                    <xsd:element name="hasOrganisationallyUniqueIdentifier"
minOccurs="1" maxOccurs="1">
                        <xsd:simpleType>
                            <xsd:restriction base="xsd:hexBinary">
                                <xsd:minLength value="3"/>
                                <xsd:maxLength value="3"/>

                            </xsd:restriction>
                        </xsd:simpleType>
                    </xsd:element>
                    <xsd:element name="hasAssigneeAddressOf" minOccurs="1"
maxOccurs="1">
                        <xsd:simpleType>
                            <xsd:restriction base="xsd:hexBinary">
                                <xsd:minLength value="3"/>
                                <xsd:maxLength value="3"/>

                            </xsd:restriction>
                        </xsd:simpleType>
                    </xsd:element>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="L_Spectrum">
        <xsd:complexContent>
            <xsd:extension base="nfp:L_LocativeEntity">
                <xsd:sequence>
                    <xsd:element name="hasFrequencyBandOf"
type="nfp:L_FrequencyBand" maxOccurs="1" minOccurs="1"/>
                    <xsd:element name="hasSpectrumFrequencyOf"
type="nfp:L_Frequency" maxOccurs="1" minOccurs="1"/>
                    <xsd:element name="isAvailableInRegionOf"
type="nfp:L_Region" maxOccurs="1" minOccurs="1"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="L_StreetDirectoryReference">
        <xsd:complexContent>
            <xsd:extension base="nfp:L_LocativeEntity">
                <xsd:sequence>
                    <xsd:element name="hasXPositionOf" type="xsd:string"
maxOccurs="1" minOccurs="1"/>
                    <xsd:element name="hasYPositionOf" type="xsd:string"
maxOccurs="1" minOccurs="1"/>
                    <xsd:element name="hasRegionOf" type="nfp:L_Region"
maxOccurs="1" minOccurs="0"/>
                    <xsd:element name="hasMapNumberOf"
type="xsd:nonNegativeInteger" maxOccurs="1" minOccurs="1"/>
                    <xsd:element name="hasReferenceTo"
```

```
type="nfp:L_StreetDirectory" maxOccurs="unbounded" minOccurs="1"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:simpleType name="L_Direction">
        <xsd:restriction base="xsd:nonNegativeInteger">
            <xsd:minExclusive value="0"/>
            <xsd:maxExclusive value="360"/>
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:simpleType name="L_Postcode">
        <xsd:restriction base="xsd:string"/>
    </xsd:simpleType>


    <xsd:simpleType name="Preposition">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="Above"/>
            <xsd:enumeration value="Below"/>
            <xsd:enumeration value="Opposite"/>
            <xsd:enumeration value="In"/>
            <xsd:enumeration value="Next"/>
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:simpleType name="L_IPAddressNumber">
        <!--
          Represents arbitrary hex-encoded binary data. A hexBinary is the
          set of finite-length sequences of binary octets. Each binary octet
          is encoded as a character tuple, consisting of two hexadecimal
          digits ([0-9a-fA-F]) representing the octet code.
        -->
        <xsd:restriction base="xsd:hexBinary">
            <xsd:maxLength value="1"/>
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:complexType name="L_FrequencyBand">
         <xsd:sequence>
             <xsd:element name="hasAllowableFrequencyFrom"
type="nfp:L_Frequency" maxOccurs="1" minOccurs="1"/>
             <xsd:element name="hasAllowableFrequencyTo"
type="nfp:L_Frequency" minOccurs="1" maxOccurs="1"/>
             <xsd:element name="hasFrequencyBandNameOf"
type="nfp:L_FrequencyBandName" minOccurs="1" maxOccurs="1"/>
         </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="L_Frequency">
         <xsd:sequence>
             <xsd:element name="hasFrequencyNumberOf"
```

```
type="nfp:L_FrequencyNumber" maxOccurs="1" minOccurs="1"/>
             <xsd:element name="hasOscillationUnitsOf"
type="nfp:L_OscillationUnits" minOccurs="1" maxOccurs="1"/>
         </xsd:sequence>
    </xsd:complexType>

    <xsd:simpleType name="L_FrequencyNumber">
        <xsd:restriction base="xsd:nonNegativeInteger"/>
    </xsd:simpleType>


    <xsd:simpleType name="L_OscillationUnits">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="GHz"/>
            <xsd:enumeration value="Mhz"/>
            <xsd:enumeration value="kHz"/>
            <xsd:enumeration value="Hz"/>
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:simpleType name="L_FrequencyBandName">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="AMRadio"/>
            <xsd:enumeration value="FMRadio"/>
            <xsd:enumeration value="CitizensBandRadio"/>
            <xsd:enumeration value="TV"/>
            <xsd:enumeration value="Microwave"/>
            <xsd:enumeration value="ShortWave"/>
            <xsd:enumeration value="Infrared"/>
            <xsd:enumeration value="RFID"/>
        </xsd:restriction>
    </xsd:simpleType>

    <!-- TODO: This is a little redundant. We should be able to determine this
from
               the type of LocativeEntity that we are dealing with. The
provision
               availability schema though requires that we are able to refer to
               the type of the LocativeEntity. -->
    <xsd:simpleType name="L_LocativeEntityType">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="Point"/>
            <xsd:enumeration value="MovingPoint"/>
            <xsd:enumeration value="Route"/>
            <xsd:enumeration value="Region"/>
            <xsd:enumeration value="Address"/>
            <xsd:enumeration value="StreetAddress"/>
            <xsd:enumeration value="PostboxAddress"/>
            <xsd:enumeration value="URI"/>
            <xsd:enumeration value="PhoneNumber"/>
            <xsd:enumeration value="IPAddress"/>
            <xsd:enumeration value="EthernetAddress"/>
            <xsd:enumeration value="Spectrum"/>
            <xsd:enumeration value="StreetDirectoryReference"/>
```

```
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:complexType name="L_StreetDirectory">
         <xsd:sequence>
             <xsd:element name="hasEditionOf" type="xsd:nonNegativeInteger"
maxOccurs="1" minOccurs="0"/>
             <xsd:element name="hasProviderOf" type="nfp:P_Provider"
maxOccurs="1" minOccurs="0"/>
             <xsd:element name="hasPublisherOf" type="nfp:P_Provider"
maxOccurs="1" minOccurs="0"/>
             <xsd:element name="hasISBNCodeOf" type="nfp:L_ISBN" maxOccurs="1"
minOccurs="1"/>
             <xsd:element name="hasPublicationTitleOf" type="xsd:string"
minOccurs="1" maxOccurs="1"/>
             <xsd:element name="hasPublicationDateOf"
type="nfp:T_TemporalDate" maxOccurs="1" minOccurs="0"/>
         </xsd:sequence>
    </xsd:complexType>

    <xsd:simpleType name="L_ISBN">
        <xsd:restriction base="xsd:string"/>
    </xsd:simpleType>

   <xsd:complexType name="Obligation">
     <xsd:sequence>
         <xsd:element name="hasObligation">
             <xsd:complexType>
                 <xsd:choice>
                   <xsd:element name="PricingObligation"
type="nfp:PricingObligation"/>
                       <xsd:element name="RelationshipObligation"
type="nfp:O_RelationshipObligation"/>
                       <xsd:element name="Membership"
type="nfp:O_Membership"/>
                       <xsd:element name="PaymentObligation"
type="nfp:O_PaymentObligation"/>
                       <xsd:element name="DeferredPaymentObligation"
type="nfp:O_DeferredPaymentObligation"/>
                  </xsd:choice>
               </xsd:complexType>
            </xsd:element>
        </xsd:sequence>
   </xsd:complexType>

   <xsd:complexType name="O_ProviderObligation">
       <xsd:sequence>
           <xsd:element name="hasObligationOf" type="nfp:Obligation"
maxOccurs="1" minOccurs="1"/>
           <xsd:element name="forProvider" type="nfp:P_Provider" maxOccurs="1"
minOccurs="1"/>
       </xsd:sequence>
    </xsd:complexType>
```

```
    <xsd:complexType name="O_Obligation"/>

    <xsd:complexType name="O_RelationshipObligation">
        <xsd:complexContent>
            <xsd:extension base="nfp:O_Obligation">
                <xsd:sequence>
         <xsd:element name="hasRelationshipConditionOf" type="nfp:Condition"
maxOccurs="unbounded" minOccurs="0"/>
         <xsd:element name="hasRelationshipDurationOf"
type="nfp:T_TemporalDuration"                maxOccurs="1" minOccurs="1"/>
         <xsd:element name="isRequiredToReceive"
type="nfp:PricingObligation" maxOccurs="1" minOccurs="0"/>
         <xsd:element name="hasAssociatedRightOf" type="nfp:Right"
maxOccurs="unbounded" minOccurs="0"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="O_Membership">
        <xsd:complexContent>
            <xsd:extension base="nfp:O_RelationshipObligation">
                <xsd:sequence>
                    <!-- Was previously O_PricingCondition. -->
         <xsd:element name="hasMembershipLevelChargeOf"
type="nfp:PricingObligation" maxOccurs="1" minOccurs="1"/>
         <xsd:element name="hasMembershipLevelOf"
type="nfp:MembershipLevelName" maxOccurs="1" minOccurs="1"/>
     </xsd:sequence>
    </xsd:extension>
</xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="MembershipLevelName">
        <xsd:simpleContent>
             <xsd:extension base="xsd:string"/>
        </xsd:simpleContent>
    </xsd:complexType>

    <xsd:complexType name="ServicePayment">
        <xsd:sequence>
            <xsd:element name="paymentObligationOf"
type="nfp:O_PaymentObligation" maxOccurs="unbounded" minOccurs="0"/>
            <xsd:element name="hasPaymentOptionOf" type="nfp:PaymentOption"
maxOccurs="unbounded" minOccurs="0"/>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="O_PaymentObligation">
        <xsd:complexContent>
            <xsd:extension base="nfp:O_Obligation">
                <xsd:sequence>
                    <xsd:element name="hasChargeOf"
type="nfp:PricingObligation" maxOccurs="1" minOccurs="0"/>
```

```
                    <xsd:element name="requiresPaymentDepositOrBondOf"
type="nfp:PricingObligation" maxOccurs="1" minOccurs="0"/>
                    <xsd:element name="requiresEstablishmentFeeOf"
type="nfp:PricingObligation" maxOccurs="1" minOccurs="0"/>
                    <xsd:element name="hasPaymentDiscountOf"
type="nfp:D_PaymentDiscount" maxOccurs="unbounded" minOccurs="0"/>

                    <xsd:element name="hasPaymentObligationConditionOf"
type="nfp:Condition" maxOccurs="unbounded" minOccurs="0"/>
                    <xsd:element name="hasInterestFreePeriodOf"
type="nfp:T_TemporalDuration" maxOccurs="1" minOccurs="0"/>
                    <xsd:element name="hasInterestCharge"
type="nfp:TemporalGranularityPricingObligation" maxOccurs="1" minOccurs="0"/>

                    <xsd:element name="hasAdministrativeCharge"
type="nfp:TemporalGranularityPricingObligation" maxOccurs="1" minOccurs="0"/>

         <xsd:element name="canBeExecutedWithInsuranceServiceOf"
type="nfp:S_Service"/>
         <xsd:element name="canBeExecutedWithEscrowServiceOf"
type="nfp:S_Service"/>
     </xsd:sequence>
    </xsd:extension>
</xsd:complexContent>
</xsd:complexType>

    <xsd:complexType name="TemporalGranularityPricingObligation">
        <xsd:sequence>
            <xsd:element name="applied" type="nfp:T_TemporalGranularity"
maxOccurs="1" minOccurs="1"/>
            <xsd:element name="of" type="nfp:PricingObligation" maxOccurs="1"
minOccurs="1"/>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="O_DeferredPaymentObligation">
        <xsd:complexContent>
            <xsd:extension base="nfp:O_PaymentObligation">
                <xsd:sequence>
                    <xsd:element name="hasDeferredPaymentConditionOf"
type="nfp:Condition" maxOccurs="unbounded" minOccurs="0"/>
                    <xsd:element name="offersDeferredPaymentPeriodOf"
type="nfp:T_TemporalDuration" maxOccurs="unbounded" minOccurs="0"/>
                    <xsd:element name="enforcesMinimumPurchaseOf"
type="nfp:P_Price" maxOccurs="unbounded" minOccurs="0"/>
                    <xsd:element name="hasDefinedPaymentScheduleOf"
type="nfp:PaymentSchedule"  maxOccurs="unbounded" minOccurs="0"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="PaymentInstrument">
        <xsd:sequence>
```

```
            <xsd:element name="supportsCurrency" type="tML-ISO-
4217:currencyCodeAlpha3Type" maxOccurs="unbounded" minOccurs="1"/>
            <xsd:element name="isIssuedIn" type="nfp:L_Region" maxOccurs="1"
minOccurs="0"/>
            <xsd:element name="isLimitedToUseIn" type="nfp:L_Region"
maxOccurs="1" minOccurs="0"/>
            <xsd:element name="hasIssuerOf" type="nfp:P_Provider"
maxOccurs="1" minOccurs="0"/>
            <xsd:element name="hasSurchargeOf">
                <xsd:complexType>
                    <xsd:choice>
                        <xsd:element name="AbsolutePriceProvider"
type="nfp:AbsolutePriceProvider"/>
                        <xsd:element name="ProportionalPriceProvider"
type="nfp:ProportionalPriceProvider"/>
                    </xsd:choice>
                </xsd:complexType>
            </xsd:element>
            <xsd:element name="supportsPaymentSchemeOf"
type="nfp:PaymentScheme" maxOccurs="1" minOccurs="0"/>
            <xsd:element name="hasPaymentInstrumentTypeOf"
type="nfp:PaymentInstrumentType" maxOccurs="1" minOccurs="1"/>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="PaymentScheme">
        <xsd:sequence>
                <xsd:element name="hasNameOf" type="xsd:string" maxOccurs="1"
minOccurs="1"/>
                <xsd:element name="controlledBy" type="nfp:P_Provider"
maxOccurs="1" minOccurs="1"/>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="AbsolutePriceProvider">
        <xsd:sequence>
            <xsd:element name="hasSurchargeOf" type="nfp:T_AbsolutePrice"
maxOccurs="1" minOccurs="1"/>
            <xsd:element name="withProvider" type="nfp:P_Provider"
maxOccurs="1" minOccurs="1"/>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="ProportionalPriceProvider">
        <xsd:sequence>
            <xsd:element name="hasSurchargeOf" type="nfp:T_ProportionalPrice"
maxOccurs="1" minOccurs="1"/>
            <xsd:element name="withProvider" type="nfp:P_Provider"
maxOccurs="1" minOccurs="1"/>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="CashInstrument">
        <xsd:complexContent>
```

```
            <xsd:extension base="nfp:PaymentInstrument"/>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="ElectronicCashInstrument">
        <xsd:complexContent>
            <xsd:extension base="nfp:CashInstrument">
                <xsd:sequence>
                    <xsd:element name="hasElectronicCashTypeOf"
type="nfp:ElectronicCashType" maxOccurs="1" minOccurs="1"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:simpleType name="ElectronicCashType">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="DirectDebit"/>
            <xsd:enumeration value="DirectTransfer"/>
            <xsd:enumeration value="DigitalCash"/>
            <xsd:enumeration value="WireTransfer"/>
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:complexType name="VoucherBasedInstrument">
        <xsd:complexContent>
            <xsd:extension base="nfp:PaymentInstrument">
                <xsd:sequence>
                    <xsd:element name="hasTemporalValidityOf"
type="nfp:TemporalEntity" maxOccurs="1" minOccurs="0"/>
                 <xsd:element name="validForRedemptionWith">
                       <xsd:complexType>
                        <xsd:choice>
                         <xsd:element name="validForRedemptionWithService"
type="nfp:S_Service"/>
                         <xsd:element name="validForRedemptionWithProvider"
type="nfp:P_Provider"/>
                           </xsd:choice>
                       </xsd:complexType>
                    </xsd:element>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="ChequeInstrument">
        <xsd:complexContent>
            <xsd:extension base="nfp:PaymentInstrument">
                <xsd:sequence>
                    <xsd:element name="hasChequeTypeOf" type="nfp:ChequeType"
maxOccurs="1" minOccurs="1"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
```

```
    </xsd:complexType>

    <xsd:simpleType name="ChequeType">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="Personal"/>
            <xsd:enumeration value="Bank"/>
            <xsd:enumeration value="Travellor"/>
            <xsd:enumeration value="Business"/>
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:complexType name="CardBasedInstrument">
        <xsd:complexContent>
            <xsd:extension base="nfp:PaymentInstrument">
                <xsd:sequence>
                    <xsd:element name="hasCardTypeOf" type="nfp:CardType"
                                 maxOccurs="1" minOccurs="1"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:simpleType name="CardType">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="Debit"/>
            <xsd:enumeration value="Credit"/>
            <xsd:enumeration value="Store"/>
            <xsd:enumeration value="Charge"/>
            <xsd:enumeration value="StoredValue"/>
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:complexType name="PaymentOption">
        <xsd:sequence>
            <xsd:element name="hasInvoiceTermsOfPaymentOf"
type="nfp:T_TemporalDuration" maxOccurs="1" minOccurs="0"/>
            <xsd:element name="hasAssociatedChargeOf"
type="nfp:PricingObligation" maxOccurs="1" minOccurs="0"/>
            <xsd:element name="limitedToRequestorsFromRegion"
type="nfp:L_Region" maxOccurs="unbounded" minOccurs="0"/>
            <xsd:element name="hasPaymentOptionConditionOf"
type="nfp:Condition" maxOccurs="unbounded" minOccurs="0"/>
            <xsd:element name="hasPaymentScheduleOf"
type="nfp:PaymentSchedule" maxOccurs="unbounded" minOccurs="0"/>
            <xsd:element name="providesTaxInvoice" type="xsd:boolean"
maxOccurs="1" minOccurs="1"/>
            <xsd:element name="isPreferredPaymentOption" type="xsd:boolean"
maxOccurs="1" minOccurs="1"/>
            <xsd:element name="supportsPaymentInstrumentAtLocations"
type="nfp:InstrumentLocations" maxOccurs="unbounded" minOccurs="1"/>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="InstrumentLocations">
```

```
        <xsd:sequence>
            <xsd:element name="canBeFulFilledUsingPaymentInstrument"
type="nfp:PaymentInstrument"  maxOccurs="1" minOccurs="1"/>
            <xsd:element name="atPaymentLocationOf" type="nfp:PaymentLocation"
maxOccurs="1" minOccurs="1"/>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="PaymentLocation">
        <xsd:sequence>
            <xsd:element name="hasLocationOf" type="nfp:LocativeEntity"
maxOccurs="1" minOccurs="1"/>
            <xsd:element name="acceptsCombinationsOfInstrumentsOf"
type="nfp:PaymentInstrumentType" maxOccurs="unbounded" minOccurs="0"/>
            <xsd:element name="requiresInPersonAttendance" type="xsd:boolean"
maxOccurs="1" minOccurs="1"/>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="PaymentSchedule">
        <xsd:sequence>
         <xsd:element name="requires">
                <xsd:complexType>
                 <xsd:choice>
                     <xsd:element name="PercentServiceProvision"
type="nfp:PercentServiceProvision"/>
                     <xsd:element name="PercentTemporalPattern"
type="nfp:PercentTemporalPattern"/>
                   </xsd:choice>
                </xsd:complexType>
            </xsd:element>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="PercentTemporalPattern">
        <xsd:sequence>
            <xsd:element name="percentage" type="nfp:Percentage" maxOccurs="1"
minOccurs="1"/>
            <xsd:element name="accordingToTemporalPattern"
type="nfp:TemporalEntity" maxOccurs="1" minOccurs="1"/>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="PercentServiceProvision">
        <xsd:sequence>
            <xsd:element name="percentage" type="nfp:Percentage" maxOccurs="1"
minOccurs="1"/>
            <xsd:element name="relationshipToServiceProvision"
type="nfp:TemporalRelationship" maxOccurs="1" minOccurs="1"/>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:simpleType name="TemporalRelationship">
        <xsd:restriction base="xsd:string">
```

```
            <xsd:enumeration value="Before(Upfront)"/>
            <xsd:enumeration value="During"/>
            <xsd:enumeration value="After"/>
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:complexType name="PaymentInstrumentType">
        <xsd:sequence>
            <xsd:element name="hasTypeOf" type="nfp:P_PaymentInstrumentType"
maxOccurs="1" minOccurs="1"/>
            <xsd:element name="hasCharacteristicOf"
type="nfp:PaymentCharacteristicAndValue" maxOccurs="unbounded" minOccurs="1"/>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:simpleType name="P_PaymentInstrumentType">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="C"/>
            <xsd:enumeration value="Q"/>
            <xsd:enumeration value="M"/>
            <xsd:enumeration value="V"/>
            <xsd:enumeration value="E"/>
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:complexType name="PaymentCharacteristicAndValue">
        <xsd:sequence>
            <xsd:element name="hasPaymentCharacteristicOf"
type="nfp:PaymentCharacteristic" maxOccurs="1" minOccurs="1"/>
            <xsd:element name="withValueOf"
type="nfp:PaymentCharacteristicValue" maxOccurs="1" minOccurs="1"/>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="PaymentCharacteristic">
        <xsd:sequence>
             <xsd:element name="hasTypeOf"
type="nfp:PaymentCharactertisticType" maxOccurs="1" minOccurs="1"/>
             <xsd:element name="hasAllowedValueOf"
type="nfp:PaymentCharacteristicValue" maxOccurs="unbounded" minOccurs="1"/>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:simpleType name="PaymentCharactertisticType">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="Acceptability"/>
            <xsd:enumeration value="Traceability"/>
            <xsd:enumeration value="Liquidity"/>
            <xsd:enumeration value="Offline"/>
            <xsd:enumeration value="Online"/>
            <xsd:enumeration value="Refutability"/>
            <xsd:enumeration value="Negotiability"/>
            <xsd:enumeration value="Expiration"/>
            <xsd:enumeration value="ProviderCoupling"/>
```

```
            <xsd:enumeration value="Transferability"/>
            <xsd:enumeration value="Security"/>
            <xsd:enumeration value="Immediacy"/>
         </xsd:restriction>
    </xsd:simpleType>

    <xsd:simpleType name="PaymentCharacteristicValue">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="High"/>
            <xsd:enumeration value="Medium"/>
            <xsd:enumeration value="Low"/>
            <xsd:enumeration value="Immediate"/>
            <xsd:enumeration value="Delayed"/>
            <xsd:enumeration value="True"/>
            <xsd:enumeration value="False"/>
            <xsd:enumeration value="Possibly"/>
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:complexType name="PricingObligation">
        <xsd:complexContent>
            <xsd:extension base="nfp:O_Obligation">
                <xsd:sequence>
                    <xsd:element name="resultsInPriceOf"
type="nfp:ServicePrice" maxOccurs="unbounded"/>
                    <xsd:element name="hasPricingObligationAvailabilityOf"
type="nfp:Availability" maxOccurs="unbounded" minOccurs="0"/>
                    <xsd:element name="hasPricingConditionOf"
type="nfp:Condition" maxOccurs="unbounded" minOccurs="0"/>
                    <xsd:element name="hasRefundConditionOf"
type="nfp:Condition" maxOccurs="unbounded" minOccurs="0"/>
                    <xsd:element name="hasRefundProcedureOf"
type="nfp:Procedure" minOccurs="0"/>
                    <xsd:element name="hasNegotiablePrice" type="xsd:boolean"
maxOccurs="1" minOccurs="1"/>
                    <xsd:element name="requiresCustomisedPricingOrQuote"
type="xsd:boolean" maxOccurs="1" minOccurs="1"/>
                    <xsd:element name="relatesTo">
                        <xsd:complexType>
                            <xsd:choice>
                                <xsd:element name="relatesToRequestTypeOf"
type="nfp:S_RequestType"/>
                                <xsd:element name="relatesToProvision"
type="xsd:boolean"/>
                            </xsd:choice>
                        </xsd:complexType>
                    </xsd:element>
                    <xsd:element name="hasAvailablePayeeDiscountOf"
type="nfp:D_PayeeDiscount" maxOccurs="unbounded" minOccurs="1"/>
                </xsd:sequence>
            </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
```

```
   <xsd:complexType name="P_Price">
       <xsd:sequence>
           <xsd:choice>
               <xsd:element name="AbsolutePrice" type="nfp:T_AbsolutePrice"/>
               <xsd:element name="ProportionalPrice"
type="nfp:T_ProportionalPrice"/>
               <xsd:element name="RangedAbsolutePrice"
type="nfp:T_RangedAbsolutePrice"/>
               <xsd:element name="RangedProportionalPrice"
type="nfp:T_RangedProportionalPrice"/>
               <xsd:element name="DynamicPrice" type="nfp:T_DynamicPrice"/>
           </xsd:choice>
       </xsd:sequence>
   </xsd:complexType>

   <xsd:complexType name="ServicePrice">
       <xsd:sequence>
           <xsd:element name="hasPrice" type="nfp:P_Price"/>
           <xsd:element name="hasPriceGranularity"
type="nfp:P_ItemGranularity" maxOccurs="unbounded" minOccurs="1"/>
           <xsd:element name="providesRightOf" type="nfp:Right"
maxOccurs="unbounded" minOccurs="1"/>
           <xsd:element name="hasPriceModifierOf" type="nfp:PriceModifier"
maxOccurs="1" minOccurs="1"/>
           <xsd:element name="hasTaxOf" maxOccurs="1" minOccurs="0">
               <xsd:complexType>
                   <xsd:choice>
                       <xsd:element name="hasInclusiveTaxOf"
type="nfp:P_TaxItem"/>
                       <xsd:element name="hasExclusiveTaxOf"
type="nfp:P_TaxItem"/>
                   </xsd:choice>
               </xsd:complexType>
           </xsd:element>
           <xsd:element name="validOnlyWithUsageOfAdditionalService"
type="nfp:S_Service" maxOccurs="unbounded" minOccurs="0"/>
      </xsd:sequence>
   </xsd:complexType>

   <xsd:complexType name="T_Price"/>
   <xsd:complexType name="T_AbsolutePrice">
        <xsd:complexContent>
            <xsd:extension base="nfp:T_Price">
                <xsd:sequence>
                    <xsd:element name="hasAmount" type="xsd:decimal"/>
                    <xsd:element name="hasCurrency" type="tML-ISO-
4217:currencyCodeAlpha3Type"/>
                </xsd:sequence>
            </xsd:extension>
       </xsd:complexContent>
   </xsd:complexType>
   <xsd:complexType name="T_ProportionalPrice">
       <xsd:complexContent>
           <xsd:extension base="nfp:T_Price">
```

```
            <xsd:sequence>
                <xsd:element name="hasPercentage" type="nfp:Percentage"/>
                <xsd:element name="hasItem" type="nfp:Item"/>

            </xsd:sequence>
          </xsd:extension>
       </xsd:complexContent>
   </xsd:complexType>
   <xsd:complexType name="T_RangedPrice">
        <xsd:complexContent>
            <xsd:extension base="nfp:T_Price"/>
        </xsd:complexContent>
   </xsd:complexType>
   <xsd:complexType name="T_RangedAbsolutePrice">
       <xsd:complexContent>
           <xsd:extension base="nfp:T_RangedPrice">
               <xsd:sequence>
                       <xsd:element name="hasRangedAbsolutePriceFrom"
type="nfp:T_AbsolutePrice"/>
                       <xsd:element name="hasRangedAbsolutePriceTo"
type="nfp:T_AbsolutePrice"/>
               </xsd:sequence>
           </xsd:extension>
       </xsd:complexContent>
   </xsd:complexType>
   <xsd:complexType name="T_RangedProportionalPrice">
       <xsd:complexContent>
           <xsd:extension base="nfp:T_RangedPrice">
               <xsd:sequence>
                   <xsd:element name="hasRangedProportionalPriceFrom"
type="nfp:T_ProportionalPrice"/>
                   <xsd:element name="hasRangedProportionalPriceTo"
type="nfp:T_ProportionalPrice"/>
               </xsd:sequence>
           </xsd:extension>
       </xsd:complexContent>
   </xsd:complexType>

   <xsd:complexType name="T_DynamicPrice">
       <xsd:complexContent>
           <xsd:extension base="nfp:T_Price">
               <xsd:sequence>
                   <xsd:element name="hasMechanismProvider"
type="nfp:P_Provider" maxOccurs="1" minOccurs="0"/>
                   <xsd:choice maxOccurs="1" minOccurs="0">

                       <xsd:element name="hasIndicativePrice-
RangedAbsolutePrice" type="nfp:T_RangedAbsolutePrice"/>
                       <xsd:element name="hasIndicativePrice-
RangedProportionalPrice" type="nfp:T_RangedProportionalPrice"/>

                   </xsd:choice>
                   <xsd:choice maxOccurs="1" minOccurs="0">
```

```
                        <xsd:element name="hasReservedPrice-
RangedAbsolutePrice" type="nfp:T_RangedAbsolutePrice"/>
                        <xsd:element name="hasReservedPrice-
RangedProportionalPrice" type="nfp:T_RangedProportionalPrice"/>

                    </xsd:choice>
                    <xsd:element name="hasMechanismType" maxOccurs="1"
minOccurs="0">
                        <xsd:simpleType>
                            <xsd:restriction base="xsd:string">
                                <xsd:enumeration value="DutchAuction"/>
                                <xsd:enumeration value="DoubleAuction"/>
                                <xsd:enumeration value="EnglishAuction"/>
                                <xsd:enumeration value="VickreyAuction"/>
                                <xsd:enumeration value="SealedBidAuction"/>

                            </xsd:restriction>
                       </xsd:simpleType>
                    </xsd:element>
                    <xsd:element name="hasMechanismCondition"
type="nfp:Condition" maxOccurs="unbounded" minOccurs="0"/>
                    <xsd:element name="hasDynamicPriceAvailability"
type="nfp:Availability" maxOccurs="unbounded" minOccurs="1"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="P_UnitOfMeasure">
        <xsd:sequence>
          <xsd:element name="hasUnitOfMeasure">
            <xsd:simpleType>
                <xsd:restriction base="xsd:string">
                    <xsd:enumeration value="person"/>
                    <xsd:enumeration value="time"/>
                    <xsd:enumeration value="weight"/>
                    <xsd:enumeration value="area"/>
                    <xsd:enumeration value="length"/>

                </xsd:restriction>
            </xsd:simpleType>
          </xsd:element>
       </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="P_Granularity">
      <xsd:sequence>
        <xsd:element name="hasGranularity">
           <xsd:simpleType>
               <xsd:restriction base="xsd:string">
                   <xsd:enumeration value="adult"/>
                   <xsd:enumeration value="child"/>
                   <xsd:enumeration value="hour"/>
                   <xsd:enumeration value="minute"/>
```

```
                <xsd:enumeration value="second"/>
                <xsd:enumeration value="night"/>
                <xsd:enumeration value="kilometre"/>
                <xsd:enumeration value="metre"/>
                <xsd:enumeration value="centimetre"/>
                <xsd:enumeration value="millimetre"/>

            </xsd:restriction>
         </xsd:simpleType>
       </xsd:element>
     </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="P_GranularityAndUnitOfMeasure">
      <xsd:sequence>
          <xsd:element name="hasUnitOfMeasure" type="nfp:P_UnitOfMeasure"/>
          <xsd:element name="hasGranularity" type="nfp:P_Granularity"/>
      </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="P_ItemGranularity">
      <xsd:sequence>
          <xsd:element name="hasCardinalityOf" type="xsd:integer"
maxOccurs="1" minOccurs="1"/>
          <xsd:element name="hasGranularityOf"
type="nfp:P_GranularityAndUnitOfMeasure" maxOccurs="unbounded" minOccurs="1"/>
          <xsd:element name="hasItemNumberOf" type="xsd:nonNegativeInteger"
maxOccurs="unbounded" minOccurs="1"/>
          <xsd:element name="relatesToItem" type="nfp:Item" maxOccurs="1"
minOccurs="0"/>
      </xsd:sequence>
  </xsd:complexType>

   <xsd:complexType name="P_TaxRateForRegion">
       <xsd:sequence>
           <xsd:element name="hasTaxRateOf" type="nfp:Percentage"/>
           <xsd:element name="hasRegionOf" type="nfp:L_Region"/>
       </xsd:sequence>
   </xsd:complexType>

   <xsd:complexType name="P_TaxItem">
       <xsd:sequence>
           <xsd:element name="hasTaxTypeOf" type="nfp:TaxType"/>
           <xsd:element name="hasTaxRateForRegionOf"
type="nfp:P_TaxRateForRegion"/>
       </xsd:sequence>
   </xsd:complexType>

  <xsd:simpleType name="Percentage">
     <xsd:restriction base="xsd:decimal">
         <xsd:totalDigits value='3'/>
         <xsd:fractionDigits value='2'/>
         <xsd:minInclusive value='0.00'/>
         <xsd:maxInclusive value='1.00'/>
```

```
      </xsd:restriction>
   </xsd:simpleType>

    <xsd:simpleType name="PriceModifier">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="Exact"/>
            <xsd:enumeration value="LimitedTo"/>
            <xsd:enumeration value="From"/>
            <xsd:enumeration value="Inclusive"/>
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:simpleType name="TaxType">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="GST"/>
            <xsd:enumeration value="VAT"/>
            <xsd:enumeration value="FederalTax"/>
            <xsd:enumeration value="StateTax"/>
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:complexType name="Service_Condition">
        <xsd:sequence>
            <xsd:element name="performsPriceMatchingOnServiceOf"
type="nfp:S_Service" maxOccurs="1" minOccurs="1"/>
            <xsd:element name="hasConditionOf" type="nfp:Condition"
maxOccurs="1" minOccurs="1"/>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="PriceMatching">
        <xsd:sequence>
            <xsd:element name="hasServiceConditionOf"
type="nfp:Service_Condition" maxOccurs="1" minOccurs="1"/>
            <xsd:element name="offersCheaperPriceBy" type="nfp:Percentage"
maxOccurs="1" minOccurs="0"/>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="Procedure">
        <xsd:simpleContent>
            <xsd:extension base="xsd:anyURI"/>
        </xsd:simpleContent>
    </xsd:complexType>

    <xsd:complexType name="P_ServiceParty"/>

    <xsd:complexType name="P_Provider">
        <xsd:complexContent>
            <xsd:extension base="nfp:P_ServiceParty">
                <xsd:sequence>
               <xsd:element name="hasProviderName" type="xsd:string"
maxOccurs="1" minOccurs="1"/>
                     <xsd:element name="hasPriceMatchingOf"
```

```
type="nfp:PriceMatching" maxOccurs="1" minOccurs="1"/>
                    <xsd:element name="hasComplianceOf"
type="nfp:AchievedCompliance" maxOccurs="unbounded" minOccurs="0"/>
                    <xsd:element name="offersPriceMatchingOf"
type="nfp:PriceMatching" maxOccurs="1" minOccurs="1"/>
                    <xsd:element name="hasProviderFeedbackOf"
type="nfp:Endorsement" maxOccurs="1" minOccurs="0"/>
                    <xsd:element name="hasMissionStatementOf"
type="nfp:Statement" maxOccurs="1" minOccurs="0"/>
                    <xsd:element name="isLegallyBoundBy"
type="nfp:Legislation" maxOccurs="unbounded" minOccurs="1"/>
                    <xsd:element name="hasYearOfInceptionOf" type="xsd:gYear"
maxOccurs="1" minOccurs="0"/>
                    <xsd:element name="hasProviderMembershipOf"
type="nfp:ProviderMembership" maxOccurs="unbounded" minOccurs="0"/>
                    <xsd:element name="hasAssociationWithOf"
type="nfp:AssociationTypeProvider" maxOccurs="unbounded" minOccurs="0"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="AssociationTypeProvider">
        <xsd:sequence>
            <xsd:element name="hasAssociationTypeOf" type="nfp:AssociationType"
                         maxOccurs="1" minOccurs="1"/>
            <xsd:element name="withProviderOf" type="nfp:P_Provider"
                         maxOccurs="1" minOccurs="1"/>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="ProviderMembership">
        <xsd:sequence>
            <xsd:element name="providerOf" type="nfp:P_Provider" maxOccurs="1"
minOccurs="1"/>
            <xsd:element name="hasMembershipOf" type="nfp:O_Membership"
maxOccurs="1" minOccurs="1"/>
            <xsd:element name="wasAchievedOn" type="nfp:T_TemporalDate"
maxOccurs="1" minOccurs="1"/>
            <xsd:element name="hasMembershipExpiryOf"
type="nfp:T_TemporalDate" maxOccurs="1" minOccurs="1"/>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="P_Requestor">
        <xsd:complexContent>
            <xsd:extension base="nfp:P_ServiceParty"/>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="AchievedCompliance">
        <xsd:sequence>
            <xsd:element name="hasComplianceOf" type="nfp:Compliance"
maxOccurs="1" minOccurs="1"/>
```

```
            <xsd:element name="hasConformanceRatingOf"
type="nfp:StandardLevelName" maxOccurs="1" minOccurs="1"/>
            <xsd:element name="wasAchievedOn" type="nfp:T_TemporalDate"
maxOccurs="1" minOccurs="1"/>
            <xsd:element name="wasVerifiedBy" type="nfp:P_Provider"
maxOccurs="1" minOccurs="1"/>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="Compliance">
        <xsd:sequence>
            <xsd:element name="achievedConformanceOfStandard"
type="nfp:Q_Standard" maxOccurs="1" minOccurs="1"/>
            <xsd:element name="forService" type="nfp:S_Service" maxOccurs="1"
minOccurs="1"/>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:simpleType name="AssociationType">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="Partner"/>
            <xsd:enumeration value="Subsidiary"/>
            <xsd:enumeration value="Owner"/>
            <xsd:enumeration value="SupplierTo"/>
            <xsd:enumeration value="Agency"/>
            <xsd:enumeration value="Division"/>
            <xsd:enumeration value="Branch"/>
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:complexType name="Q_Standard">
        <xsd:sequence>
         <xsd:element name="hasPublisherOf">
             <xsd:complexType>
                 <xsd:choice>
                       <xsd:element name="publisher" type="nfp:P_Provider"/>
                       <xsd:element name="serviceIndustry"
type="unspsc:UNSPSC_Type"/>
       </xsd:choice>
       </xsd:complexType>
         </xsd:element>
          <xsd:element name="supercedes" type="nfp:Q_Standard"
maxOccurs="unbounded" minOccurs="0"/>
           <xsd:element name="hasCoverageOf" type="nfp:L_Region"
maxOccurs="unbounded" minOccurs="0"/>
          <xsd:element name="isAvailableAt" type="nfp:LocativeEntity"
maxOccurs="1" minOccurs="0"/>
          <xsd:element name="hasSupportedConformanceLevelOf"
type="nfp:StandardLevelName" maxOccurs="unbounded" minOccurs="0"/>
          <xsd:element name="hasStatusOf" type="nfp:StatusName"
maxOccurs="unbounded" minOccurs="0"/>
          <xsd:element name="hasPublicationDateOf" type="nfp:T_TemporalDate"
maxOccurs="1" minOccurs="1"/>         <xsd:element name="hasTitleOf"
type="nfp:TitleName" maxOccurs="1" minOccurs="1"/>
```

```
        <xsd:element name="hasAuthorOf" type="nfp:Author" maxOccurs="1"
minOccurs="0"/>
        <xsd:element name="hasVersionNumberOf" type="xsd:decimal"
maxOccurs="1" minOccurs="0"/>
     </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="Rating">
     <xsd:sequence>
        <xsd:element name="hasNameOf" type="xsd:string" maxOccurs="1"
minOccurs="1"/>
        <xsd:element name="hasDescriptionOf" type="xsd:string" maxOccurs="1"
minOccurs="0"/>
        <xsd:element name="hasRatingOf">
            <xsd:complexType>
                <xsd:choice>
                      <xsd:element name="indicator" type="nfp:Indicator"/>
                      <xsd:element name="ranking" type="nfp:Ranking"/>
       </xsd:choice>
       </xsd:complexType>
      </xsd:element>
          <xsd:element name="hasRatingValueOf" type="nfp:RatingValue"
maxOccurs="1" minOccurs="0"/>
   </xsd:sequence>
   </xsd:complexType>

    <xsd:complexType name="RatingValue">
     <xsd:sequence>
        <xsd:element name="isApplicableToRegionOf" type="nfp:L_Region"
maxOccurs="1" minOccurs="0"/>
        <xsd:element name="hasRatingValueOf">
            <xsd:complexType>
                <xsd:choice>
                      <xsd:element name="hasPriceValueOf" type="nfp:P_Price"/>
                      <xsd:element name="hasNumericValueOf"
type="xsd:decimal"/>
                      <xsd:element name="hasPercentageValueOf"
type="nfp:Percentage"/>
                      <xsd:element name="hasStringValueOf" type="xsd:string"/>
</xsd:choice>
  </xsd:complexType>
    </xsd:element>
    <xsd:element name="hasPercentageChangeOf" type="nfp:Percentage"
maxOccurs="1" minOccurs="0"/>
</xsd:sequence>
   </xsd:complexType>

    <xsd:complexType name="AssessmentStandard">
        <xsd:complexContent>
            <xsd:extension base="nfp:Q_Standard">
            <xsd:sequence>
             <xsd:element name="hasParticipantOf" type="nfp:P_Provider"
maxOccurs="unbounded" minOccurs="0"/>
             <xsd:element name="hasAssessmentStandardOf">
```

```
                <xsd:complexType>
                   <xsd:choice>
                          <xsd:element name="Benchmark" type="nfp:Benchmark"/>
                          <xsd:element name="RankingScheme"
type="nfp:RankingScheme"/>
             </xsd:choice>
      </xsd:complexType>
        </xsd:element>
              <xsd:element name="hasRatingValueOf" type="nfp:RatingValue"
maxOccurs="1" minOccurs="0"/>       </xsd:sequence>
      </xsd:extension>
      </xsd:complexContent>
   </xsd:complexType>

   <xsd:complexType name="RankingScheme">
      <xsd:complexContent>
            <xsd:extension base="nfp:AssessmentStandard">
    <xsd:sequence>
         <xsd:element name="hasRankingOf" type="nfp:Ranking"
maxOccurs="unbounded" minOccurs="1"/>
    </xsd:sequence>
    </xsd:extension>
    </xsd:complexContent>
   </xsd:complexType>

   <xsd:complexType name="Ranking">
     <xsd:sequence>
         <xsd:element name="isValidForPeriodOf" type="nfp:T_TemporalDuration"
maxOccurs="1" minOccurs="0"/>
         <xsd:element name="hasRankingPositionOf"
type="xsd:nonNegativeInteger" maxOccurs="1" minOccurs="1"/>
      </xsd:sequence>
   </xsd:complexType>

   <xsd:complexType name="Benchmark">
        <xsd:complexContent>
            <xsd:extension base="nfp:AssessmentStandard">
             <xsd:sequence>
          <xsd:element name="hasIndicatorOf" type="nfp:Indicator"
maxOccurs="unbounded" minOccurs="1"/>
       </xsd:sequence>
</xsd:extension>
  </xsd:complexContent>
   </xsd:complexType>

   <xsd:complexType name="Indicator">
     <xsd:sequence>
         <xsd:element name="isApplicableIn" type="nfp:L_Region" maxOccurs="1"
minOccurs="0"/>
         <xsd:element name="isApplicableDuring" type="nfp:TemporalEntity"
maxOccurs="1" minOccurs="0"/>
   </xsd:sequence>
   </xsd:complexType>
```

```
   <xsd:simpleType name="StandardLevelName">
    <xsd:restriction base="xsd:string"/>
   </xsd:simpleType>

   <xsd:simpleType name="StatusName">
    <xsd:restriction base="xsd:string"/>
   </xsd:simpleType>

   <xsd:simpleType name="TitleName">
    <xsd:restriction base="xsd:string"/>
   </xsd:simpleType>

   <xsd:simpleType name="Author">
       <xsd:restriction base="xsd:string"/>
   </xsd:simpleType>

   <xsd:simpleType name="QualityDimension">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="Reliability"/>
            <xsd:enumeration value="Responsiveness"/>
            <xsd:enumeration value="Efficiency"/>
        </xsd:restriction>
   </xsd:simpleType>

   <xsd:complexType name="AccumulatedReward">
     <xsd:sequence>
         <xsd:element name="rewardsSchemeOf" type="nfp:S_Service"
maxOccurs="1" minOccurs="1"/>
         <xsd:element name="providesAccumulationDuring"
type="nfp:TemporalEntity" maxOccurs="1" minOccurs="1"/>
         <xsd:element name="underRewardsAccumulationCondition"
type="nfp:Condition" maxOccurs="1" minOccurs="1"/>
     </xsd:sequence>
   </xsd:complexType>

  <xsd:complexType name="AccumulatedPriceReward">
    <xsd:sequence>
      <xsd:element name="accumulatedRewardOf" type="nfp:AccumulatedReward"
maxOccurs="1" minOccurs="1"/>
      <xsd:element name="hasRewardPointsOf" type="nfp:RewardPoint"/>
    </xsd:sequence>
  </xsd:complexType>

   <xsd:complexType name="RedeemableReward">
     <xsd:sequence>
         <xsd:element name="serviceOf" type="nfp:S_Service" maxOccurs="1"
minOccurs="1"/>
         <xsd:element name="isRewardsSchemeFor" type="nfp:S_Service"
maxOccurs="1" minOccurs="1"/>
         <xsd:element name="thatAllowsRedemptionDuring"
type="nfp:TemporalEntity" maxOccurs="1" minOccurs="1"/>
     </xsd:sequence>
   </xsd:complexType>
```

```
  <xsd:complexType name="RedeemablePaymentReward">
    <xsd:sequence>
      <xsd:element name="redeemableRewardOf" type="nfp:RedeemableReward"
maxOccurs="1" minOccurs="1"/>
      <xsd:element name="requiresPointsOf" type="nfp:RewardPoint"/>
      <xsd:element name="hasRedemptionConditionOf" type="nfp:Condition"
maxOccurs="unbounded" minOccurs="1"/>
      <xsd:element name="satisfiesPaymentFor" type="nfp:ServicePayment"
maxOccurs="1" minOccurs="1"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:simpleType name="RewardPoint">
      <xsd:restriction base="xsd:nonNegativeInteger"/>
  </xsd:simpleType>

  <xsd:complexType name="Right">
       <xsd:sequence>
       <xsd:element name="hasNameOf" type="xsd:string" maxOccurs="1"
minOccurs="1"/>
         <xsd:element name="hasRight">
             <xsd:complexType>
                 <xsd:choice>
                  <xsd:element name="RightOfRefusal"
type="nfp:RightOfRefusal"/>
                  <xsd:element name="RightOfExtension"
type="nfp:RightOfExtension"/>
                  <xsd:element name="RightOfAccess" type="nfp:RightOfAccess"/>

                  <xsd:element name="RightOfRecourse"
type="nfp:RightOfRecourse"/>                                    <xsd:element
name="RightOfSuspension" type="nfp:RightOfSuspension"/>
                  <xsd:element name="RightOfTermination"
type="nfp:RightOfTermination"/>
                  <xsd:element name="RightOfPrivacy" type="nfp:RightOfPrivacy"/>
                  <xsd:element name="RightOfWarranty"
type="nfp:RightOfWarranty"/>                                    <xsd:element
name="RightOfDisclosure" type="nfp:RightOfDisclosure"/>
                  <xsd:element name="RegisterIPRight"
type="nfp:RegisteredIPRight"/>
  <xsd:element name="RightOfLiabilityLimitation"
type="nfp:RightOfLiabilityLimitation"/>
       </xsd:choice>
       </xsd:complexType>
    </xsd:element>
    <xsd:element name="hasTemporalValidityOf" type="nfp:TemporalEntity"
maxOccurs="unbounded" minOccurs="1"/>
    <xsd:element name="isGrantedTo" type="nfp:P_ServiceParty"
maxOccurs="unbounded" minOccurs="1"/>
  </xsd:sequence>
   </xsd:complexType>

   <xsd:complexType name="R_Right"/>
```

```
   <xsd:complexType name="RightOfWarranty">
        <xsd:complexContent>
            <xsd:extension base="nfp:R_Right">
        <xsd:sequence>
     <xsd:element name="hasWarrantiedItems" type="nfp:WarrantiedItem"
maxOccurs="unbounded" minOccurs="1"/>
<xsd:element name="fulfilsWarranty" type="nfp:P_Provider" maxOccurs="1"
minOccurs="1"/>
<xsd:element name="hasWarrantyConditionOf" type="nfp:Condition"
maxOccurs="unbounded" minOccurs="1"/>
<xsd:element name="hasWarrantyProcedureAvailabilityOf"
type="nfp:WarrantyProcedureAvailability" maxOccurs="unbounded" minOccurs="1"/>
    </xsd:sequence>
        </xsd:extension>
        </xsd:complexContent>
   </xsd:complexType>

<xsd:complexType name="WarrantyProcedureAvailability">
<xsd:sequence>
<xsd:element name="hasWarrantyProcedureOf" type="nfp:Procedure"
maxOccurs="1" minOccurs="1"/>
<xsd:element name="hasAvailabilityOf" type="nfp:Availability" maxOccurs="1"
minOccurs="1"/>
</xsd:sequence>
</xsd:complexType>

   <xsd:complexType name="WarrantiedItem">
       <xsd:sequence>
<xsd:element name="isProvidedForItem" type="nfp:Item" maxOccurs="1"
minOccurs="1"/>
         <xsd:element name="forWarrantyPeriodOf" type="nfp:T_TemporalDuration"
maxOccurs="1" minOccurs="1"/>
       </xsd:sequence>
   </xsd:complexType>

   <xsd:complexType name="RightOfAccess">
        <xsd:complexContent>
            <xsd:extension base="nfp:R_Right">
    <xsd:sequence>
<xsd:element name="hasAccessTypeResourceOf" type="nfp:AccessTypeResource"
maxOccurs="unbounded" minOccurs="1"/>
          <xsd:element name="hasObligationTemporalDurationOf"
type="nfp:ObligationTemporalDuration" maxOccurs="unbounded" minOccurs="0"/>
     </xsd:sequence>
        </xsd:extension>
       </xsd:complexContent>
   </xsd:complexType>

   <xsd:complexType name="ObligationTemporalDuration">
       <xsd:sequence>
<xsd:element name="incursObligationOf" type="nfp:Obligation" maxOccurs="1"
minOccurs="1"/>
         <xsd:element name="forExclusiveAccessDurationOf"
type="nfp:T_TemporalDuration" maxOccurs="1" minOccurs="1"/>
```

```
       </xsd:sequence>
   </xsd:complexType>

   <xsd:complexType name="AccessTypeResource">
       <xsd:sequence>
<xsd:element name="isAvailableForTemporalDurationOf"
type="nfp:T_TemporalDuration" maxOccurs="1" minOccurs="1"/>
         <xsd:element name="underConditionOf" type="nfp:Condition"
maxOccurs="1" minOccurs="1"/>
       </xsd:sequence>
   </xsd:complexType>

   <xsd:complexType name="Resource">
       <xsd:sequence>
      <xsd:element name="hasResourceLocationOf" type="nfp:LocativeEntity"
maxOccurs="1" minOccurs="0"/>
         <xsd:element name="hasResourceNameOf" type="xsd:string" maxOccurs="1"
minOccurs="1"/>
         <xsd:element name="hasResourceTypeOf" type="nfp:ResourceType"
maxOccurs="1" minOccurs="1"/>
       </xsd:sequence>
   </xsd:complexType>

   <xsd:simpleType name="ResourceType">
        <xsd:restriction base="xsd:string">
             <xsd:enumeration value="IntellectualProperty"/>
             <xsd:enumeration value="Time"/>
             <xsd:enumeration value="Information"/>
             <xsd:enumeration value="Design"/>
             <xsd:enumeration value="Person"/>
             <xsd:enumeration value="Facility"/>
        </xsd:restriction>
   </xsd:simpleType>

   <xsd:simpleType name="AccessType">
        <xsd:restriction base="xsd:string">
             <xsd:enumeration value="Exclusive"/>
             <xsd:enumeration value="Shared"/>
             <xsd:enumeration value="Restricted"/>
             <xsd:enumeration value="Prohibited"/>
        </xsd:restriction>
   </xsd:simpleType>

   <xsd:complexType name="RightOfExtension">
        <xsd:complexContent>
            <xsd:extension base="nfp:R_Right">
        <xsd:sequence>
   <xsd:element name="incursObligationOf" type="nfp:Obligation"
maxOccurs="unbounded" minOccurs="0"/>
             <xsd:element name="isInitiatedUsingProcedureOf"
type="nfp:Procedure" maxOccurs="unbounded" minOccurs="0"/>
             <xsd:element name="hasTemporalDurationConditionOf"
type="nfp:TemporalDurationCondition" maxOccurs="unbounded" minOccurs="1"/>
        </xsd:sequence>
```

```
     </xsd:extension>
      </xsd:complexContent>
   </xsd:complexType>

   <xsd:complexType name="TemporalDurationCondition">
       <xsd:sequence>
    <xsd:element name="isAvailableForTemporalDurationOf"
type="nfp:T_TemporalDuration" maxOccurs="1" minOccurs="1"/>
        <xsd:element name="underConditionOf" type="nfp:Condition"
maxOccurs="1" minOccurs="1"/>
       </xsd:sequence>
   </xsd:complexType>

   <xsd:complexType name="RightOfRefusal">
        <xsd:complexContent>
            <xsd:extension base="nfp:R_Right">
     <xsd:sequence>
            <xsd:element name="refusalIsAllowed">
              <xsd:complexType>
                  <xsd:choice>
                        <xsd:element name="isDiscretionary"
type="xsd:boolean"/>
                        <xsd:element name="isAllowedUnderConditionOf"
type="nfp:Condition"/>
         </xsd:choice>
         </xsd:complexType>
           </xsd:element>
            <xsd:element name="isAdministeredUnderRefusalProcedureOf"
type="nfp:Procedure" maxOccurs="1" minOccurs="0"/>
            <xsd:element name="hasAppealOf" type="nfp:Appeal"
maxOccurs="unbounded" minOccurs="0"/>
    </xsd:sequence>
    </xsd:extension>
   </xsd:complexContent>
   </xsd:complexType>

   <xsd:complexType name="Appeal">
       <xsd:sequence>
          <xsd:element name="hasAppealsProcedureOf" type="nfp:Procedure"
maxOccurs="1" minOccurs="1"/>
         <xsd:element name="withAvailability" type="nfp:Availability"
maxOccurs="1" minOccurs="1"/>
     </xsd:sequence>
   </xsd:complexType>

   <xsd:complexType name="RightOfPrivacy">
        <xsd:complexContent>
            <xsd:extension base="nfp:R_Right">
     <xsd:sequence>
          <xsd:element name="isAvailableFor">
              <xsd:complexType>
                  <xsd:choice>
                        <xsd:element name="isBoundByLegislation"
type="nfp:Legislation"/>
```

```
                        <xsd:element name="hasPrivacyStatementOf"
type="nfp:Statement"/>
      </xsd:choice>
</xsd:complexType>
            </xsd:element>
          <xsd:element name="hasPrivacyPolicyTypeConditionOf"
type="nfp:PrivacyPolicyTypeCondition" maxOccurs="unbounded" minOccurs="1"/>
           <xsd:element name="hasAvailabilityOf" type="nfp:Availability"
maxOccurs="unbounded" minOccurs="0"/>
    </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
   </xsd:complexType>

   <!-- TODO: Show the relationship between the disclosed item and the
        provider it is shared with. -->
   <xsd:complexType name="DisclosedItem">
       <xsd:sequence>
<xsd:element name="hasDisclosedItemDurationsOf"
type="nfp:DisclosedItemDurations" maxOccurs="unbounded" minOccurs="0"/>
         <xsd:element name="isSharedWithProvider" type="nfp:P_Provider"
maxOccurs="unbounded" minOccurs="0"/>
       </xsd:sequence>
   </xsd:complexType>

   <xsd:complexType name="DisclosedItemDurations">
       <xsd:sequence>
         <xsd:element name="holdsRequestorInformationOf"
type="nfp:DisclosureItem" maxOccurs="1" minOccurs="1"/>
         <xsd:element name="forDurationOf" type="nfp:T_TemporalDuration"
maxOccurs="1" minOccurs="1"/>
       </xsd:sequence>
   </xsd:complexType>

   <xsd:complexType name="PrivacyPolicyTypeCondition">
       <xsd:sequence>
         <xsd:element name="hasPolicyOfType" type="nfp:PrivacyPolicyType"
maxOccurs="1" minOccurs="1"/>
         <xsd:element name="withPrivacyConditionOf" type="nfp:Condition"
maxOccurs="1" minOccurs="1"/>
   </xsd:sequence>
   </xsd:complexType>

   <xsd:simpleType name="PrivacyPolicyType">
        <xsd:restriction base="xsd:string">
             <xsd:enumeration value="Collection"/>
             <xsd:enumeration value="Access"/>
             <xsd:enumeration value="Storage"/>
             <xsd:enumeration value="Alteration"/>
        </xsd:restriction>
   </xsd:simpleType>

   <xsd:complexType name="RightOfDisclosure">
        <xsd:complexContent>
```

```
            <xsd:extension base="nfp:R_Right">
     <xsd:sequence>
          <xsd:element name="capturesInformationRelatingTo"
type="nfp:DisclosureItem" maxOccurs="unbounded" minOccurs="1"/>
    </xsd:sequence>
    </xsd:extension>
    </xsd:complexContent>
   </xsd:complexType>

<xsd:simpleType name="DisclosureItem">
<xsd:restriction base="nfp:Item"/>
</xsd:simpleType>

   <xsd:complexType name="RightOfTermination">
        <xsd:complexContent>
            <xsd:extension base="nfp:R_Right">
     <xsd:sequence>
            <xsd:element name="hasTerminationProcedureOf"
type="nfp:Procedure" maxOccurs="unbounded" minOccurs="0"/>
            <xsd:element name="hasNotificationPeriodOf"
type="nfp:T_TemporalDuration" maxOccurs="1" minOccurs="0"/>
            <xsd:element name="incursObligationOf" type="nfp:Obligation"
maxOccurs="unbounded" minOccurs="0"/>
            <xsd:element name="resultsFromFailureToMeetObligationOf"
type="nfp:Obligation" maxOccurs="unbounded" minOccurs="0"/>
            <xsd:element name="hasTerminationConditionOf"
type="nfp:Condition" maxOccurs="unbounded" minOccurs="0"/>
            <xsd:element name="hasTerminationTypeOf"
type="nfp:TerminationType" maxOccurs="1" minOccurs="1"/>
    </xsd:sequence>
    </xsd:extension>
   </xsd:complexContent>
   </xsd:complexType>

   <xsd:simpleType name="TerminationType">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="ProviderInitiated"/>
            <xsd:enumeration value="RequestorInitiated"/>
        </xsd:restriction>
   </xsd:simpleType>

   <xsd:complexType name="RightOfSuspension">
        <xsd:complexContent>
            <xsd:extension base="nfp:R_Right">
     <xsd:sequence>
          <xsd:element name="hasSuspensionConditionOf" type="nfp:Condition"
maxOccurs="unbounded" minOccurs="0"/>
          <xsd:element name="hasResumptionConditionOf" type="nfp:Condition"
maxOccurs="unbounded" minOccurs="0"/>
          <xsd:element name="hasSuspensionObligationOf" type="nfp:Obligation"
maxOccurs="1" minOccurs="0"/>
          <xsd:element name="hasResumptionObligationOf" type="nfp:Obligation"
maxOccurs="1" minOccurs="0"/>
          <xsd:element name="hasMaximumNumberOfSuspensionsOf"
```

```
type="xsd:nonNegativeInteger" maxOccurs="1" minOccurs="0"/>
          <xsd:element name="hasMinimumSuspensionPeriodOf"
type="nfp:T_TemporalDuration" maxOccurs="1" minOccurs="0"/>
          <xsd:element name="hasMaximumSuspensionPeriodOf"
type="nfp:T_TemporalDuration" maxOccurs="1" minOccurs="0"/>
          <xsd:element name="hasMaximumAggregatedSuspensionPeriodOf"
type="nfp:T_TemporalDuration" maxOccurs="1" minOccurs="0"/>
          <xsd:element name="hasSuspensionProcedureOf" type="nfp:Procedure"
maxOccurs="1" minOccurs="0"/>
          <xsd:element name="hasResumptionProcedureOf" type="nfp:Procedure"
maxOccurs="1" minOccurs="0"/>
          <xsd:element name="hasSuspensionAvailabilityOf"
type="nfp:Availability" maxOccurs="1" minOccurs="0"/>
    </xsd:sequence>
    </xsd:extension>
     </xsd:complexContent>
   </xsd:complexType>

   <xsd:complexType name="RightOfCoolingOffPeriod">
        <xsd:complexContent>
            <xsd:extension base="nfp:R_Right">
     <xsd:sequence>
          <xsd:element name="isAvailableFor">
                <xsd:complexType>
                  <xsd:choice>
                    <xsd:element name="isAvailableForDurationAfterRequest"
type="nfp:T_TemporalDuration"/>
                    <xsd:element name="isAvailableForDurationIntoProvision"
type="nfp:T_TemporalDuration"/>
   </xsd:choice>
        </xsd:complexType>
         </xsd:element>
          <xsd:element name="hasAnnulmentProcedureOf" type="xsd:anyURI"
maxOccurs="1" minOccurs="0"/>
          <xsd:element name="hasCoolingOffConditionOf" type="nfp:Condition"
maxOccurs="unbounded" minOccurs="0"/>
    </xsd:sequence>
</xsd:extension>
  </xsd:complexContent>
   </xsd:complexType>

   <xsd:complexType name="RightOfLiabilityLimitation">
        <xsd:complexContent>
            <xsd:extension base="nfp:R_Right">
     <xsd:sequence>
          <xsd:element name="hasLiabilityLimitConditionOf"
type="nfp:Condition" maxOccurs="unbounded" minOccurs="0"/>
    </xsd:sequence>
    </xsd:extension>
    </xsd:complexContent>
   </xsd:complexType>

   <xsd:complexType name="RightOfRecourse">
        <xsd:complexContent>
```

```
            <xsd:extension base="nfp:R_Right">
     <xsd:sequence>
          <xsd:element name="hasPeriodOfAvailabilityForRecourseOf"
type="nfp:T_TemporalDuration" maxOccurs="1" minOccurs="1"/>
          <xsd:element name="hasRecourseProcedureOf" type="nfp:Procedure"
maxOccurs="1" minOccurs="0"/>
          <xsd:element name="mediated">
              <xsd:complexType>
                  <xsd:choice>
                       <xsd:element name="isMediatedThrough"
type="nfp:P_Provider"/>
                       <xsd:element name="isNotMediated" type="xsd:boolean"/>
         </xsd:choice>
         </xsd:complexType>
         </xsd:element>
          <xsd:element name="hasAdministeredJurisdictionOf"
type="nfp:Jurisdiction"
                      maxOccurs="1" minOccurs="1"/>
    </xsd:sequence>
    </xsd:extension>
    </xsd:complexContent>
   </xsd:complexType>

   <!-- TODO: There are multiple forms of Jurisdiction that could apply
        here. Should the hasLegislationOf fact type be unbounded. -->
   <xsd:complexType name="Jurisdiction">
     <xsd:sequence>
         <xsd:element name="hasLocationOf" type="nfp:LocativeEntity"
                     maxOccurs="1" minOccurs="1"/>
         <xsd:element name="hasLegislationOf" type="nfp:Legislation"
                     maxOccurs="1" minOccurs="1"/>
   </xsd:sequence>
   </xsd:complexType>

   <xsd:complexType name="Legislation">
     <xsd:sequence>
         <xsd:element name="isSupersededBy" type="nfp:Legislation"
                     maxOccurs="unbounded" minOccurs="0"/>
         <xsd:element name="hasAmendmentOf" type="nfp:Legislation"
                     maxOccurs="unbounded" minOccurs="0"/>
         <xsd:element name="hasLegislationNameOf" type="xsd:string"
                     maxOccurs="1" minOccurs="1"/>
         <xsd:element name="hasYearOfIntroductionOf" type="xsd:gYear"
                     maxOccurs="1" minOccurs="1"/>
   </xsd:sequence>
   </xsd:complexType>

    <xsd:complexType name="IdentificationRequirement">
        <xsd:sequence>
            <xsd:element name="requiresCollectiveIdentificationPointsOf"
                        type="xsd:nonNegativeInteger" maxOccurs="1"
       minOccurs="0"/>
            <xsd:element name="acceptsIdentificationOf"
                        type="nfp:AcceptableIdentification"
```

```
 maxOccurs="unbounded" minOccurs="1"/>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="AcceptableIdentification">
        <xsd:sequence>
<xsd:element name="hasIdentificationType" type="nfp:IdentificationType"
 maxOccurs="1" minOccurs="1"/>
<xsd:element name="isMandatory" type="xsd:boolean"
 maxOccurs="1" minOccurs="1"/>
<xsd:element name="hasAssignedIdentificationPointsOf"
 type="xsd:nonNegativeInteger"
 maxOccurs="1" minOccurs="1"/>
  </xsd:sequence>
    </xsd:complexType>

    <!-- Indicative enumeration of values. -->
    <xsd:simpleType name="IdentificationType">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="Licence"/>
            <xsd:enumeration value="CreditCard"/>
            <xsd:enumeration value="Passport"/>
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:complexType name="Confidentiality">
        <xsd:sequence>
            <xsd:element name="hasConfidentialityConditionOf"
          type="nfp:Condition" maxOccurs="unbounded" minOccurs="0"/>
            <xsd:element name="hasConfidentialityStatementOf"
 type="nfp:Statement" maxOccurs="unbounded" minOccurs="0"/>
       <xsd:element name="hasConfidentialityOf">
              <xsd:complexType>
                <xsd:choice>
                   <xsd:element name="ConfidentialityMechanism"
    type="nfp:ConfidentialityMechanism"/>
                   <xsd:element name="ConfidentialityAgreement"
    type="nfp:ConfidentialityAgreement"/>
                 </xsd:choice>
              </xsd:complexType>
           </xsd:element>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="ConfidentialityMechanism">
        <xsd:complexContent>
            <xsd:extension base="nfp:Confidentiality">
            <xsd:sequence>
              <xsd:element name="supportsEncryptionTechniqueOf"
       type="nfp:EncryptionTechnique"
              maxOccurs="1" minOccurs="1"/>
           </xsd:sequence>
        </xsd:extension>
        </xsd:complexContent>
```

```
   </xsd:complexType>

   <xsd:complexType name="ConfidentialityAgreement">
       <xsd:complexContent>
           <xsd:extension base="nfp:Confidentiality">
           <xsd:sequence>
         <xsd:element name="hasValidityPeriodOf" type="nfp:T_TemporalInterval"
                  maxOccurs="1" minOccurs="0"/>
    <xsd:element name="isControlledWithin" type="nfp:Jurisdiction"
                     maxOccurs="1" minOccurs="0"/>
           </xsd:sequence>
       </xsd:extension>
       </xsd:complexContent>
   </xsd:complexType>

   <xsd:complexType name="EncryptionTechnique">
       <xsd:sequence>
           <xsd:element name="hasNameOf" type="xsd:string" maxOccurs="1"
      minOccurs="1"/>
           <xsd:element name="hasKeyLengthOf" type="xsd:nonNegativeInteger"
                        maxOccurs="1" minOccurs="1"/>
           <xsd:element name="hasDetailsOf" type="nfp:Q_Standard"
                        maxOccurs="1" minOccurs="1"/>
       </xsd:sequence>
   </xsd:complexType>

   <xsd:complexType name="TemporalEntity">
       <xsd:sequence>
          <xsd:element name="hasTemporalValue">
              <xsd:complexType>
                  <xsd:choice>
                    <xsd:element name="TemporalTime" type="nfp:T_TemporalTime"/>
             <xsd:element name="OrdinalDate" type="nfp:T_OrdinalDate"/>
     <xsd:element name="WeekDate" type="nfp:T_WeekDate"/>
                      <xsd:element name="CalendarMonthDate"
           type="nfp:T_CalendarMonthDate"/>
     <xsd:element name="CalendarDate" type="nfp:T_CalendarDate"/>
     <xsd:element name="RecurringDailyTimeInAWeek"
      type="nfp:T_RecurringDailyTimeInAWeek"/>
     <xsd:element name="RecurringDailyTimeInAMonth"
type="nfp:T_RecurringDailyTimeInAMonth"/>
     <xsd:element name="RecurringDayOfWeekInMonthTime"
type="nfp:T_RecurringDayOfWeekInMonthTime"/>
     <xsd:element name="RecurringDayOfMonthTime"
        type="nfp:T_RecurringDayOfMonthTime"/>
     <xsd:element name="AnchoredPointinTime"
type="nfp:T_AnchoredPointinTime"/>
                      <xsd:element name="TemporalDuration"
type="nfp:T_TemporalDuration"/>
                      <xsd:element name="TemporalInterval"
type="nfp:T_TemporalInterval"/>
 </xsd:choice>
   </xsd:complexType>
     </xsd:element>
```

```
 </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="T_TemporalEntity">
<xsd:sequence>
    <xsd:element name="hasTemporalEntityConditionOf" type="nfp:Condition"/>
          <xsd:element name="hasCommonNameOf" type="nfp:T_TemporalCommon"
                       maxOccurs="unbounded" minOccurs="0"/>
      </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="T_TemporalCommon">
  <xsd:sequence>
           <xsd:element name="hasTemporalCommonNameOf" type="xsd:string"
maxOccurs="1" minOccurs="1"/>
           <xsd:element name="appliesToLocation" type="nfp:L_LocativeEntity"
maxOccurs="1" minOccurs="0"/>
      </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="T_TemporalTime">
      <xsd:complexContent>
          <xsd:extension base="nfp:T_TemporalEntity">
              <xsd:sequence>
                  <xsd:element name="hasHourOf" type="nfp:T_Hours"/>
                  <xsd:element name="hasMinutesOf" type="nfp:T_Minutes"/>
                  <xsd:element name="hasSecondsOf" type="nfp:T_Seconds"/>
                  <xsd:element name="hasTimezoneOf" type="nfp:T_Timezone"/>
              </xsd:sequence>
          </xsd:extension>
      </xsd:complexContent>
  </xsd:complexType>

  <xsd:complexType name="T_Timezone">
      <xsd:sequence>
          <xsd:element name="hasHourOf">
              <xsd:simpleType>
                  <xsd:restriction base="xsd:nonNegativeInteger">
                      <xsd:minExclusive value="0"/>
                      <xsd:maxExclusive value="12"/>
                  </xsd:restriction>
              </xsd:simpleType>
          </xsd:element>
          <xsd:element name="hasMinutesOf" type="nfp:T_Minutes"/>
      </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="T_TemporalDate">
      <xsd:complexContent>
          <xsd:extension base="nfp:T_TemporalEntity">
              <xsd:sequence>
                  <xsd:element name="hasYearOf" type="xsd:gYear"/>
              </xsd:sequence>
          </xsd:extension>
```

```
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="T_OrdinalDate">
    <xsd:complexContent>
        <xsd:extension base="nfp:T_TemporalDate">
            <xsd:sequence>
                <xsd:element name="hasDayOfYearOf">
                    <xsd:simpleType>
                        <xsd:restriction base="xsd:nonNegativeInteger">
                            <xsd:minExclusive value="1"/>
                            <xsd:maxExclusive value="366"/>
                        </xsd:restriction>
                    </xsd:simpleType>
                </xsd:element>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="T_WeekDate">
    <xsd:complexContent>
        <xsd:extension base="nfp:T_TemporalDate">
            <xsd:sequence>
                <xsd:element name="hasWeekNumberOf">
                    <xsd:simpleType>
                        <xsd:restriction base="xsd:nonNegativeInteger">
                            <xsd:minExclusive value="1"/>
                            <xsd:maxExclusive value="52"/>
                        </xsd:restriction>
                    </xsd:simpleType>
                </xsd:element>
                <xsd:element name="hasDayOfWeekNumberOf"
                             type="nfp:T_DayOfWeekNumber"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="T_CalendarMonthDate">
    <xsd:complexContent>
        <xsd:extension base="nfp:T_TemporalDate">
            <xsd:sequence>
                <xsd:element name="hasMonthNumberOf"
                             type="nfp:T_MonthNumber"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="T_CalendarDate">
    <xsd:complexContent>
        <xsd:extension base="nfp:T_CalendarMonthDate">
            <xsd:sequence>
```

```
                <xsd:element name="hasDayOfMonthNumberOf"
                             type="nfp:T_DayOfMonthNumber"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="T_DayOfWeekNumber">
    <xsd:sequence>
        <xsd:element name="hasDayOfWeekNumberOf">
            <xsd:simpleType>
                <xsd:restriction base="xsd:nonNegativeInteger">
                    <xsd:minExclusive value="1"/>
                    <xsd:maxExclusive value="7"/>
                </xsd:restriction>
            </xsd:simpleType>
        </xsd:element>
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="T_MonthNumber">
    <xsd:sequence>
        <xsd:element name="hasMonthNumberOf" type="xsd:gMonth"/>
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="T_DayOfMonthNumber">
    <xsd:sequence>
        <xsd:element name="hasDayOfMonthNumberOf" type="xsd:gDay"/>
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="T_RecurringDailyTimeInAWeek">
    <xsd:complexContent>
        <xsd:extension base="nfp:T_TemporalTime">
            <xsd:sequence>
                <xsd:element name="hasDayOfWeekNumberOf"
                             type="nfp:T_DayOfWeekNumber"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="T_RecurringDailyTimeInAMonth">
    <xsd:complexContent>
        <xsd:extension base="nfp:T_TemporalTime">
            <xsd:sequence>
                <xsd:element name="hasMonthNumberOf"
                             type="nfp:T_MonthNumber"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>
```

```
   <!-- From a modelling perspective we have not completed this to my liking.
        We are unable to reproduce multiple inheritance in XML Schema and we
        are therefore inheriting the properties of one supertype (i.e.
        T_RecurringDailyTimeInAWeek) and referring to the other supertype using
        an instance of the type (i.e. nfp:T_RecurringDailyTimeInAMonth).
     Justin O'Sullivan (9th August 2006).
    -->
   <xsd:complexType name="T_RecurringDayOfWeekInMonthTime">
       <xsd:complexContent>
           <xsd:extension base="nfp:T_RecurringDailyTimeInAWeek">
               <xsd:sequence>
                   <xsd:element name="hasOccurrenceNumberForDayOfWeekOf"
                                type="xsd:integer"/>
       <xsd:element name="hasRecurringDailyTimeInAMonth"
  type="nfp:T_RecurringDailyTimeInAMonth"/>
               </xsd:sequence>
           </xsd:extension>
       </xsd:complexContent>
   </xsd:complexType>

   <xsd:complexType name="T_RecurringDayOfMonthTime">
       <xsd:complexContent>
           <xsd:extension base="nfp:T_RecurringDailyTimeInAMonth">
               <xsd:sequence>
                   <xsd:element name="hasDayOfMonthNumberOf"
                                type="nfp:T_DayOfMonthNumber"/>
               </xsd:sequence>
           </xsd:extension>
       </xsd:complexContent>
   </xsd:complexType>

   <xsd:complexType name="T_AnchoredPointinTime">
       <xsd:complexContent>
           <xsd:extension base="nfp:T_TemporalTime">
               <xsd:choice>
                   <xsd:element name="hasOrdinalDateOf" type="nfp:T_OrdinalDate"/>
                   <xsd:element name="hasWeekDateOf" type="nfp:T_WeekDate"/>
                   <xsd:element name="hasCalendarDateOf" type="nfp:T_CalendarDate"/>
               </xsd:choice>
           </xsd:extension>
       </xsd:complexContent>
   </xsd:complexType>

   <xsd:complexType name="T_TemporalDuration">
       <xsd:sequence>
           <xsd:element name="hasCardinalityOf" type="xsd:integer"/>
           <xsd:element name="hasTemporalGranularityOf"
type="nfp:T_TemporalGranularity"
                    maxOccurs="unbounded" minOccurs="1"/>
           <xsd:element name="hasAlternativeRepresentationAsTemporalGranularityOf"
                     type="nfp:T_UserDefinedTemporalGranularity"
maxOccurs="unbounded" minOccurs="0"/>
       </xsd:sequence>
```

```
   </xsd:complexType>

   <xsd:complexType name="T_TemporalGranularity">
       <xsd:sequence>
         <xsd:element name="hasTemporalGranularity">
              <xsd:complexType>
                  <xsd:choice>
                      <xsd:element name="StandardTemporalGranularity"
        type="nfp:T_StandardTemporalGranularity"/>
                      <xsd:element name="UserDefinedTemporalGranularity"
type="nfp:T_UserDefinedTemporalGranularity"/>
                  </xsd:choice>
              </xsd:complexType>
          </xsd:element>
      </xsd:sequence>
   </xsd:complexType>

   <xsd:complexType name="T_UserDefinedTemporalGranularity">
       <xsd:complexContent>
           <xsd:extension base="nfp:T_TemporalGranularity">
               <xsd:sequence>
                   <xsd:element name="hasUserDefinedGranularityNameOf"
  type="xsd:string" maxOccurs="1" minOccurs="1"/>
               </xsd:sequence>
           </xsd:extension>
       </xsd:complexContent>
   </xsd:complexType>

   <xsd:complexType name="T_StandardTemporalGranularity">
       <xsd:complexContent>
           <xsd:extension base="nfp:T_TemporalGranularity">
               <xsd:sequence>
                   <xsd:element name="hasStandardGranularityNameOf"
  maxOccurs="1" minOccurs="1">
                       <xsd:simpleType>
                          <xsd:restriction base="xsd:string">
       <xsd:enumeration value="Hour"/>
       <xsd:enumeration value="Minute"/>
       <xsd:enumeration value="Second"/>
       <xsd:enumeration value="Day"/>
       <xsd:enumeration value="Week"/>
       <xsd:enumeration value="Month"/>
       <xsd:enumeration value="Year"/>
                           </xsd:restriction>
                       </xsd:simpleType>
                   </xsd:element>
               </xsd:sequence>
           </xsd:extension>
       </xsd:complexContent>
   </xsd:complexType>

   <!-- The use of the type nfp:T_TemporalTime in the entities -
  StartAndEnd, StartAndDuration and DurationAndEnd allows for a standard
      time (8pm AEST) or an anchored point in time (8pm AEST 26th November 2005).
```

```
    -->
   <xsd:complexType name="StartAndEnd">
       <xsd:sequence>
           <xsd:element name="hasStartTimeOf" type="nfp:T_TemporalTime"/>
           <xsd:element name="hasEndTimeOf" type="nfp:T_TemporalTime"/>
       </xsd:sequence>
   </xsd:complexType>

   <xsd:complexType name="StartAndDuration">
       <xsd:sequence>
           <xsd:element name="hasStartTimeOf" type="nfp:T_TemporalTime"/>
           <xsd:element name="hasDurationOf" type="nfp:T_TemporalDuration"/>
       </xsd:sequence>
   </xsd:complexType>

   <xsd:complexType name="DurationAndEnd">
       <xsd:sequence>
           <xsd:element name="hasDurationOf" type="nfp:T_TemporalDuration"/>
           <xsd:element name="hasEndTimeOf" type="nfp:T_TemporalTime"/>
       </xsd:sequence>
   </xsd:complexType>

   <xsd:complexType name="T_TemporalInterval">
       <xsd:complexContent>
           <xsd:extension base="nfp:T_TemporalEntity">
               <xsd:sequence>
                   <xsd:choice>
                       <xsd:element name="StartAndEnd" type="nfp:StartAndEnd"/>
                       <xsd:element name="StartAndDuration" type="nfp:StartAndDuration"/>
                       <xsd:element name="DurationAndEnd" type="nfp:DurationAndEnd"/>
                   </xsd:choice>
                   <xsd:element name="hasTemporalIntervalOperationsOf"
  type="nfp:TemporalIntervalOperation"
                                maxOccurs="unbounded" minOccurs="1"/>
                   <xsd:element name="hasIntervalTemporalEntityOccurrenceOf"
        type="nfp:IntervalTemporalEntityOccurrence"
                                maxOccurs="unbounded" minOccurs="0"/>
               </xsd:sequence>
           </xsd:extension>
       </xsd:complexContent>
   </xsd:complexType>

   <xsd:complexType name="TemporalIntervalOperation">
       <xsd:sequence>
           <xsd:element name="hasOperationTypeOf" type="nfp:OperationType"
maxOccurs="1" minOccurs="1"/>
           <xsd:element name="usingIntervalOf" type="nfp:T_TemporalInterval"/>
       </xsd:sequence>
   </xsd:complexType>

   <xsd:complexType name="IntervalTemporalEntityOccurrence">
       <xsd:sequence>
           <xsd:element name="hasOperationTypeOf" type="nfp:OperationType"
            maxOccurs="1" minOccurs="1"/>
```

```
          <xsd:element name="usingTemporalEntityOf" type="nfp:TemporalEntity"
maxOccurs="1" minOccurs="1"/>
          <xsd:element name="withOccurrenceNumberOf" type="xsd:nonNegativeInteger"
maxOccurs="1" minOccurs="1"/>
       </xsd:sequence>
   </xsd:complexType>

   <xsd:simpleType name="OperationType">
       <xsd:restriction base="xsd:string">
           <xsd:enumeration value="Refinement"/>
           <xsd:enumeration value="Exception"/>
       </xsd:restriction>
   </xsd:simpleType>

   <xsd:simpleType name="T_Hours">
       <xsd:restriction base="xsd:integer">
           <xsd:minExclusive value="0"/>
           <xsd:maxExclusive value="23"/>
       </xsd:restriction>
   </xsd:simpleType>

   <xsd:simpleType name="T_Minutes">
        <xsd:restriction base="xsd:nonNegativeInteger">
            <xsd:minExclusive value="0"/>
            <xsd:maxExclusive value="59"/>
        </xsd:restriction>
   </xsd:simpleType>

   <xsd:simpleType name="T_Seconds">
       <xsd:restriction base="xsd:nonNegativeInteger">
           <xsd:minExclusive value="0"/>
           <xsd:maxExclusive value="59"/>
       </xsd:restriction>
   </xsd:simpleType>

   <xsd:complexType name="Endorsement">
       <xsd:sequence>
          <xsd:element name="hasEndorsementOf">
              <xsd:complexType>
                  <xsd:choice>
                    <xsd:element name="internallyManagedEndorsement"
                                 type="nfp:InternallyManagedEndorsement"/>
                    <xsd:element name="externallyManagedEndorsement"
                                 type="nfp:ExternallyManagedEndorsement"/>
 </xsd:choice>
   </xsd:complexType>
     </xsd:element>
 </xsd:sequence>
   </xsd:complexType>

   <xsd:complexType name="InternallyManagedEndorsement">
       <xsd:sequence>
           <xsd:element name="hasPartyThatProvidesEndorsementOf"
type="nfp:P_ServiceParty" maxOccurs="1" minOccurs="1"/>
```

```
          <xsd:element name="hasEndorsementReceivedOn"
  type="nfp:T_AnchoredPointinTime" maxOccurs="1"
minOccurs="1"/>
     <xsd:element name="hasRatingOf" type="nfp:Rating" maxOccurs="1"
minOccurs="1"/>
       <xsd:element name="hasCommentOf" type="xsd:string" maxOccurs="1"
minOccurs="1"/>
       </xsd:sequence>
   </xsd:complexType>

   <xsd:complexType name="ExternallyManagedEndorsement">
       <xsd:sequence>
           <xsd:element name="isManagedExternallyAt" type="xsd:anyURI"
maxOccurs="1" minOccurs="1"/>
       </xsd:sequence>
   </xsd:complexType>

   <xsd:element name="Service" type="nfp:S_Service"/>

   <xsd:complexType name="S_Service">
     <xsd:sequence>
         <xsd:element name="providesCapabilityOf" type="nfp:Capability"
     maxOccurs="1" minOccurs="1"/>
         <xsd:element name="hasServiceName" type="xsd:string" maxOccurs="1"
     minOccurs="1"/>
         <xsd:element name="isOfferedByProvider" type="nfp:P_Provider"
     maxOccurs="unbounded" minOccurs="1"/>
         <xsd:element name="operatesInIndustry" type="unspsc:UNSPSC_Type"
     maxOccurs="unbounded" minOccurs="0"/>
          <xsd:element name="hasObligationForAllRequestorsOf"
     type="nfp:Obligation" maxOccurs="unbounded" minOccurs="0"/>
          <xsd:element name="hasProviderObligationOf" type="nfp:O_ProviderObligation"
     maxOccurs="unbounded" minOccurs="1"/>
          <xsd:element name="hasRequestAvailabilityOf" type="nfp:A_RequestAvailability"
     maxOccurs="unbounded" minOccurs="0"/>
         <xsd:element name="hasProvisionAvailabilityOf"
     type="nfp:A_ProvisionAvailability" maxOccurs="unbounded"
     minOccurs="0"/>
         <xsd:element name="hasAvailabilePriceRewardOf"
     type="nfp:AccumulatedPriceReward" maxOccurs="unbounded"
     minOccurs="0"/>
         <xsd:element name="hasYearOfInceptionOf" type="xsd:gYear"
                      maxOccurs="1" minOccurs="0"/>
         <xsd:element name="hasServiceFeedbackOf" type="nfp:Endorsement"
                      maxOccurs="1" minOccurs="0"/>
         <xsd:element name="capturesInteractionHistory" type="xsd:boolean"
                      maxOccurs="1" minOccurs="1"/>
         <xsd:element name="supportsUserProfile" type="xsd:boolean"
                      maxOccurs="1" minOccurs="1"/>
         <xsd:element name="hasUserProfileConfigurationLocationOf"
     type="nfp:LocativeEntity" maxOccurs="1" minOccurs="1"/>
         <xsd:element name="hasQualityEndorsementOf"
     type="nfp:QualityEndorsement" maxOccurs="unbounded" minOccurs="0"/>
         <xsd:element name="hasAssessmentOf" type="nfp:Assessment"
```

```
                    maxOccurs="unbounded" minOccurs="0"/>
   </xsd:sequence>
   </xsd:complexType>

   <xsd:complexType name="S_RequestType">
       <xsd:sequence>
           <xsd:element name="hasRequestType">
               <xsd:simpleType>
                   <xsd:restriction base="xsd:string">
                       <xsd:enumeration value="Capability"/>
                       <xsd:enumeration value="IssueResolution"/>
                       <xsd:enumeration value="Feedback"/>
                       <xsd:enumeration value="Information"/>
                   </xsd:restriction>
               </xsd:simpleType>
           </xsd:element>
       </xsd:sequence>
   </xsd:complexType>

   <xsd:simpleType name="Item">
        <xsd:restriction base="xsd:anyURI"/>
   </xsd:simpleType>

   <xsd:complexType name="QualityEndorsement">
       <xsd:sequence>
            <xsd:element name="hasFeedbackRelatingTo" type="nfp:QualityDimension"
                        maxOccurs="1" minOccurs="1"/>
            <xsd:element name="of" type="nfp:Endorsement" maxOccurs="1"
          minOccurs="1"/>
       </xsd:sequence>
   </xsd:complexType>

   <xsd:complexType name="QualityAssessment">
       <xsd:sequence>
           <xsd:element name="hasAssessmentOf" type="nfp:QualityDimension"
                       maxOccurs="1" minOccurs="1"/>
           <xsd:element name="forRating" type="nfp:Rating" maxOccurs="1"
minOccurs="1"/>
           <xsd:element name="for" type="nfp:AssessmentStandard"
maxOccurs="1" minOccurs="1"/>
       </xsd:sequence>
   </xsd:complexType>

   <xsd:complexType name="Assessment">
       <xsd:sequence>
           <xsd:element name="hasQualityAssessmentOf" type="nfp:QualityAssessment"
                       maxOccurs="1" minOccurs="1"/>
           <xsd:element name="wasAchievedOn" type="nfp:T_TemporalDate"
maxOccurs="1" minOccurs="1"/>
           <xsd:element name="hasRatingValueOf" type="nfp:RatingValue"
maxOccurs="1" minOccurs="1"/>
           <xsd:element name="verifiedIndependentlyBy" type="nfp:P_Provider"
maxOccurs="1" minOccurs="0"/>
        </xsd:sequence>
```

```
   </xsd:complexType>
</xsd:schema>
```

Bibliography

[1] R. Akkiraju, J. Farrell, J. Miller, M. Nagarajan, M.-T. Schmidt, A. Sheth, and K. Verma. Web Service Semantics - WSDL-S, 2005. Available from `http://www.w3.org/Submission/WSDL-S/`, accessed on 13-Mar-2006.

[2] J. F. Allen. Maintaining Knowledge about Temporal Intervals. *Communications of the ACM*, 26(11):832–843, 1983.

[3] J. F. Allen. Planning as Temporal Reasoning. In J. Allen, R. Fikes, and E. Sandewall, editors, *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning (KR'91) - Reprinted*, pages 3–14, Cambridge, Massachusetts, USA, 1991. Morgan Kaufmann Publishers.

[4] J. F. Allen. Time and Time Again: The Many Ways to Represent Time. *International Journal of Intelligent Systems*, 6(4):341–356, 1991.

[5] A. Ankolekar, M. Burstein, J. R. Hobbs, O. Lassila, D. L. Martin, S. A. McIlraith, S. Narayanan, M. Paolucci, T. Payne, K. Sycara, and H. Zeng. DAML-S: Semantic Markup for Web Services. In I. F. Cruz, S. Decker, J. Euzenat, and D. L. McGuinness, editors, *Proceedings of the First International Semantic Web Working Symposium (SWWS): Infrastructure and Applications for the Semantic Web*, pages 411–430, Stanford University, California, USA, 2001.

[6] C. Aurrecoechea, A. T. Campbell, and L. Hauw. A Survey of QoS Architectures. *ACM/Springer-Verlag Multimedia Systems Journal (Special Issue on QoS Architecture)*, 6(3):138–151, 1998.

[7] S. Bajaj, D. Box, D. Chappell, F. Curbera, G. Daniels, P. Hallam-Baker, M. Hondo, C. Kaler, D. Langworthy, A. Nadalin, N. Nagaratnam, H. Prafullchandra, C. von Riegen, D. Roth, J. S. (Editor), C. Sharp, J. Shewchuk, A. Vedamuthu, Ü. Yalcinalp, and D. Orchard. Web Services Policy 1.2 - Framework (WS-Policy), 2006. Available from `http://www.w3.org/Submission/2006/SUBM-WS-Policy-20060425/`, accessed on 07-Aug-2006.

[8] S. Bajaj, D. Box, D. Chappell, F. Curbera, G. Daniels, P. Hallam-Baker, M. Hondo, C. Kaler, H. Maruyama, A. Nadalin, D. Orchard, H. Prafullchandra, C. von Riegen, D. Roth, J. Schlimmer, C. Sharp, J. Shewchuk,

A. Vedamuthu, and Ü. Yalcinalp. Web Services Policy 1.2 - Attachment (WS-PolicyAttachment), 2006. Available from `http://www.w3.org/Submission/2006/SUBM-WS-PolicyAttachment-20060425/`, accessed on 07-Aug-2006.

[9] T. Berners-Lee, R. T. Fielding, and L. Masinter. Uniform Resource Identifiers (URI): Generic Syntax, 1998. Available from `http://www.ietf.org/rfc/rfc2396.txt`, accessed on 19-Jun-2001.

[10] C. Bettini, X. S. Wang, and S. Jajodia. A General Framework and Reasoning Models for Time Granularity. In L. Chittaro, S. D. Goodwin, H. J. Hamilton, and A. Montanari, editors, *Proceedings of the 3rd International Workshop on Temporal Representation and Reasoning (TIME'96)*, pages 104–111, Key West, Florida, USA, 1996. IEEE Computer Society Press.

[11] L. Bird, A. Goodchild, and T. A. Halpin. Object Role Modelling and XML-Schema. In A. H. F. Laender, S. W. Liddle, and V. C. Storey, editors, *19th International Conference on Conceptual Modeling (ER 2000)*, pages 309 – 322, Salt Lake City, Utah, USA, 2000. Springer.

[12] A. Bloesch and T. Halpin. ConQuer: A Conceptual Query Language. In *15th International Conference on Conceptual Modeling*, volume 1157, pages 121–133. Springer LNCS, 1996.

[13] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, F. Yergeau, and J. Cowan. Extensible Markup Language (XML) 1.1, 2004. Available from `http://www.w3.org/TR/2004/REC-xml11-20040204/`, accessed on 14-Jun-2006.

[14] P. Brittenham. An overview of the Web Services Inspection Language: Distributed Web service discovery useing WS-Inspection documents. Technical report, IBM Corporation, November 2001. Available from `ftp://www6.software.ibm.com/software/developer/library/ws-wsilover.pdf`, accessed on 02-Nov-2001.

[15] J. d. Bruijn, C. Bussler, D. Fensel, M. Kifer, J. Kopecky, R. Lara, E. Oren, A. Polleres, and M. Stollberg. Web Services Modeling Ontology (WSMO) - Final Draft 13th April 2005, 2005. Available from `http://www.wsmo.org/TR/d2/v1.2/`, accessed on 20-May-2005.

[16] J. d. Bruijn, H. Lausen, R. Krummenacher, A. Polleres, L. Predoiu, M. Kifer, and D. Fensel. The Web Service Modeling Language WSML - WSML Final Draft 5th October 2005, 2005. Available from `http://www.wsmo.org/TR/d16/d16.1/v0.21/`, accessed on 21-Jun-2006.

[17] C. Bussler. A Minimal Triple Space Computing Architecture. In *2nd WSMO Implementation Workshop*, Innsbruck, Austria, 2005. Digital Enterprise Research Institute.

[18] W. Caelli, D. Longley, and M. Shain. *Information Security Handbook*. Macmillan Publishers, New York, New York, USA, 1991.

[19] A. Campbell, G. Coulson, and D. Hutchison. A Quality of Service Architecture. *ACM SIGCOMM Computer Communication Review*, 24(2):6–27, 1994.

[20] F. Casati, S. Ilnicki, L. Jin, V. Krishnamoorthy, and M.-C. Shan. Adaptive and Dynamic Service Composition in eFlow. Technical Report HPL-2000-39, Hewlett Packard Software Technology Laboratory, Palo Alto, California, USA, March 2000. Available from `http://www.hpl.hp.com/techreports/2000/HPL-2000-39.html`, accessed on 27-Dec-2000.

[21] H. Chen, D. Chakraborty, L. Xu, A. Joshi, and T. Finin. Service Discovery in the Future Electronic Market. In *Workshop on Knowledge-based Electronic Markets (KBEM'00) - Proceedings of the 17th National Conference on Artificial Intelligence (AAAI-00)*, pages 1–6, Austin, Texas, USA, 2000. AAAI Press/MIT Press.

[22] R. Chinnici, J.-J. Moreau, A. Ryman, and S. Weerawarana. Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language, 2006. Available from `http://www.w3.org/TR/2006/CR-wsdl20-20060327`, accessed on 25-Jun-2006.

[23] E. Christensen, G. Meredith, F. Curbera, and S. Weerawarana. Web Services Description Language (WSDL) 1.1. W3C Note, Ariba, Microsoft and IBM Corporation, March 2001. Available from `http://www.w3.org/TR/2001/NOTE-wsdl-20010315`, accessed on 26-Jun-2001.

[24] L. Chung. Non-Functional Requirements for Information System Design. In R. Andersen, J. A. Bubenkor, and A. Sølvberg, editors, *Proceedings of the 3rd International Conference on Advanced Information Systems Engineering - CAiSE'91*, Lecture Notes in Computer Science, pages 5–30, Trodheim, Norway, 1991. Springer-Verlag.

[25] R. Clarke. Human Identification in Information Systems: Management Challenges and Public Policy Issues. *Information Technology & People*, 7(4):6 – 37, 1994.

[26] J. Clifford, C. Dyreson, T. Isakowitz, C. S. Jensen, and R. Snodgrass. On the Semantics of "Now" in Databases. *ACM Transactions on Database Systems*, 22(2):171–214, 1997.

[27] A. G. Cohn and S. M. Hazarika. Qualitative Spatial Representation and Reasoning: An Overview. *Fundamenta Informaticae*, 46(1-2):2–32, 2001.

[28] D. Corporation, I. Corporation, and X. Corporation. The Ethernet: a local area network, data link layer and physical layer specification, 30th September 1980.

[29] S. E. Czerwinski, B. Y. Zhao, T. D. Hodes, A. D. Joseph, and R. H. Katz. An Architecture for a Secure Service Discovery Service. In T. Imielinski and M. Steenstrup, editors, *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom '99)*, pages 24–35, Seattle, Washington, USA, 1999. Association for Computing Machinery.

[30] R. Davis, H. Shrobe, and P. Szolovits. What is a Knowledge Representation? *AI Magazine*, 14(1):17–33, 1993.

[31] K. Decker, M. Williamson, and K. Sycara. Matchmaking and Brokering. In *Proceedings of the 2nd International Conference on Multi-Agent Systems (ICMAS-96)*, pages 1–19, Kyoto, Japan, 1996. MIT Press.

[32] Defense Advanced Research Projects Agency. Internet Protocol, 1981. Available from `http://www.freesoft.org/CIE/RFC/791/index.htm`, accessed on 22-May-2002.

[33] M. Dumas, J. O'Sullivan, M. Heravizadeh, D. Edmond, and A. ter Hofstede. Towards a Semantic Framework for Service Description. In R. Meersman, K. Aberer, and T. Dillon, editors, *Proceedings of the 9th International Federation for Information Processing (IFIP) Conference on Database Semantics - Semantic Issues in e-Commerce Systems*, volume 239 of *IFIP International Federation for Information Processing*, pages 277–291, Hong Kong, China, 2001. Kluwer Academic Publishers.

[34] Dun and Bradstreet. Duns Number Information, 2004. Available from `https://eupdate.dnb.com/dunsnumberinfo.html`, accessed on 18-Nov-2004.

[35] A. Durante, D. Bell, L. Goldstein, J. Gustafson, and H. Kuno. A Model for the E-Service Marketplace. Technical Report HPL-2001-17, Hewlett Packard Laboratories, Palo Alto, California, USA, February 2000. Available from `http://www.hpl.hp.com/techreports/2000/HPL-2000-17.pdf`, access on 31-Aug-2001.

[36] ebXML Technical Architecture Project Team. ebXML Technical Architecture Specification v1.0.4, 16th February 2001 2001. Available from `http://www.ebxml.org/specs/ebTA.pdf`, accessed on 10-Mar-2006.

[37] ESRI. GIS Dictionary, 2004. Available from `http://support.esri.com/index.cfm?fa=knowledgebase.gisDictionary.gatewa%y`, accessed on 08-Jul-2004.

[38] FAR Laboratories. The International GPS Global Positioning System Waypoint Registry, 2006. Available from `http://www.waypoint.org/`, accessed on 09-Oct-2006.

[39] Federal Government of Australia. AusTender - The Australian Government Tender System, 2005. Available from `http://www.tenders.gov.au/`, accessed on 01-Aug-2005.

[40] D. Fensel and C. Bussler. The Web Service Modeling Framework WSMF, 2002. Available from `http://www.cs.vu.nl/~dieter/wsmf/wsmf.paper.pdf`, accessed on 07-Mar-2002.

[41] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. RFC2616 - Hypertext Transfer Protocol – HTTP/1.1, 1999. Available from `http://www.w3.org/Protocols/rfc2616/rfc2616.html`, accessed on 21-Nov-2004.

[42] R. T. Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. Phd thesis, University of California, Irvine, 2000.

[43] A. S. Fotheringham and M. Wegener, editors. *Spatial Models and GIS: New Potential and New Models*, volume 7 of *GISData*. Taylor and Francis Inc., London, England, 2000.

[44] Google. Google Maps, 2006. Available from `http://maps.google.com/`, accessed on 09-Oct-2006.

[45] K. Gottschalk, S. Graham, H. Kreger, and J. Snell. Introduction to Web services architecture. *IBM Systems Journal*, 41(2):178–198, 2002.

[46] F. Griffel, M. Boger, H. Weinreich, W. Lamersdorf, and M. Merz. Electronic Contracting with COSMOS - How to Establish, Negotiate and Execute Electronic Contracts on the Internet. In Z. M. C. Kobryn and C. Atkinson, editors, *Proceedings of the IEEE/OMG 2nd International Enterprise Distributed Object Computing Workshop (EDOC '98)*, La Jolla, California, USA, 1998. IEEE Publishing.

[47] M. Gudgin, M. Hadley, N. Mendelsohn, J.-J. Moreau, and H. F. Nielsen. SOAP Version 1.2 Part 1: Messaging Framework, 2003. Available from `http://www.w3.org/TR/2003/REC-soap12-part0-20030624/`, accessed on 14-Jun-2006.

[48] T. Halpin. *Information Modeling and Relational Databases : From Conceptual Analysis to Logical Design*. Morgan Kaufmann Publishers, San Francisco, 2001.

[49] International Organization for Standardization. ISO 8601:2000 Representation of dates and times, 2000. Available from `http://www.iso.ch`, accessed on 07-Jun-2001.

[50] International Organization for Standardization. ISO 4217:2001 Codes for the representation of currencies and funds, 2001. Available from `http://www.iso.ch`, accessed on 10-Aug-2006.

[51] IP Australia. Designs, 2004. Available from `http://www.ipaustralia.gov.au/design/what_index.shtml`, accessed on 16-Nov-2004.

[52] M. Jarrar, J. Demey, and R. Meersman. On Using Conceptual Data Modeling for Ontology Engineering. *Journal on Data Semantics*, 2800:185–207, 2003.

[53] N. R. Jennings, T. J. Norman, P. Faratin, P. O'Brien, and B. Odgers. Autonomous Agents for Business Process Management. *Journal of Applied Artificial Intelligence*, 14(2):145–189, 2000.

[54] C. S. Jensen, J. Clifford, S. K. Gadia, A. Segev, and R. T. Snodgrass. A Glossary of Temporal Database Concepts. *SIGMOD Record*, 21(3):35–43, 1992.

[55] D. Kuebler and W. Eibach. Metering and accounting for Web services, July 2001. Available from `http://www.ibm.com/developerworks/library/ws-maws/?n-ws-7191`, accessed on 31-Jul-2001.

[56] A. Lazcano, G. Alonso, H. Schuldt, and C. Schuler. The WISE Approach to Electronic Commerce. *International Journal of Computer Systems Science & Engineering, Special issue on Flexible Workflow Technology Driving the Networked Economy*, 15(5):343–355, 2000.

[57] D. Lehmann. Classes of service under competition and technological change: a model for the dynamics of the Internet? Technical Report Number TR-2000-42, Hebrew University, Jerusalem, Israel, October 2000. Available from `http://www.cs.huji.ac.il/~lehmann/papers/economics/classes_bw.ps`, accessed on 19-Jun-2001.

[58] R. C. Lewis and B. H. Booms. The Marketing Aspects of Service Quality. In L. Berry, G. Shostack, and G. Upah, editors, *Emerging Perspectives in Services Marketing*, pages 99–104. American Marketing Association, Chicago, Illinois, USA, 1983.

[59] H. Ludwig, A. Keller, A. Dan, R. P. King, and R. Franck. Web Service Level Agreement (WSLA) Language Specification, 2003. Available from `http://www.research.ibm.com/wsla/WSLASpecV1-20030128.pdf`, accessed on 14-Jul-2003.

[60] J. K. MacKie-Mason and K. White. Evaluating and Selecting Digital Payment Mechanisms. In G. L. Rosston and D. Waterman, editors, *Interconnection and the Internet. Selected Papers from the 1996 Telecommunications Policy Research Conference*, pages 113–134. Lawrence Erlbaum, Mahwah, New Jersey, USA, 1997.

[61] M. Marazakis, D. Papadakis, and C. Nikolaou. The Aurora Architecture for Developing Network-Centric Applications by Dynamic Composition of Services. Technical Report TR97-0213, Institute of Computer Science, Foundation for Research and Technology, Heraklion, Greece, December

1997. Available from http://citeseer.nj.nec.com/marazakis97aurora.html, accessed on 25-May-2001.

[62] S. P. Marsh. *Formalising Trust as a Computational Concept*. PhD Thesis, University of Stirling, Stirling, Scotland, 1994.

[63] Michelin. The Michelin Guide France, 2004. Available from http://www.viamichelin.com, accessed on 10-Aug-2004.

[64] A. C. Myers and B. Liskov. A Decentralized Model for Information Flow Control. In *Proceedings of the 16th ACM Symposium on Operating Systems Principles*, pages 129–142, Saint Malo, France, 1997. ACM Press.

[65] A. Nadalin, C. Kaler, P. Hallem-Baker, and R. Monzillo. Web Services Security: SOAP Message Security 1.0 (WS-Security 2004), 2003. Available from http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-se%curity-1.0.pdf, accessed on 13-Jul-2004.

[66] P. Oaks, A. H. M. ter Hofstede, and D. Edmond. Capabilities: Describing What Services Can Do. In M. E. Orlowska, S. Weerawarana, M. P. Papazoglou, and J. Yang, editors, *First International Conference on Service-Oriented Computing (ICSOC)*, volume 2910 of *Lecture Notes in Computer Science*, pages 1–16, Trento, Italy, 2003. Springer.

[67] OASIS Universal Business Language (UBL) Technical Committee. Universal Business Language 1.0, 15th September 2004 2004. Available from http://docs.oasis-open.org/ubl/cd-UBL-1.0/, accessed on 20-Feb-2006.

[68] J. O'Sullivan, D. Edmond, and A. ter Hofstede. What's in a service?: Towards accurate description of non-functional service properties. *Distributed and Parallel Databases Journal - Special Issue on E-Services*, 12(2-3):117–133, 2002.

[69] J. O'Sullivan, D. Edmond, and A. H. ter Hofstede. service-description.com, 2005. Available from http://www.service-description.com/, accessed on 15-Feb-2005.

[70] J. O'Sullivan, D. Edmond, and A. H. M. ter Hofstede. Formal description of non-functional service properties. Technical FIT-TR-2005-01, Queensland University of Technology, Brisbane, 9th February 2005. Available from http://www.citi.qut.edu.au/about/research_pubs/technical/non-funct%ional.jsp, accessed on 15-Feb-2005.

[71] J. O'Sullivan, D. Edmond, and A. H. M. ter Hofstede. The Price Of Services. In F. Casati, P. Traverso, and B. Benatallah, editors, *Proceedings of the The Third International Conference on Service-Oriented Computing (ICSOC05)*, pages 564–569, Amsterdam, The Netherlands, 2005. Springer Verlag.

[72] J. O'Sullivan, D. Edmond, and A. H. M. ter Hofstede. Two main challenges in service description: Web service tunnel vision and Semantic myopia. In *W3C Workshop on Frameworks for Semantics in Web Services*, Innsbruck, Austria, 2005.

[73] OWL-S Coalition. OWL-S Web Service Ontology, 2004. Available from `http://www.daml.org/services/owl-s/1.1/`, accessed on 21-Nov-2004.

[74] A. Parasuraman, V. A. Zeithaml, and L. L. Berry. SERVQUAL: A Multiple-Item Scale for Measuring Consumer Perceptions of Service Quality. *Journal of Retailing*, 64(1):12–40, 1988.

[75] J. L. A. Peiro, N. Asokan, M. Steiner, and M. Waidner. Designing a generic payment service. *IBM Systems Journal*, 37(1):72–88, 1998.

[76] D. Pfoser and C. S. Jensen. Capturing the Uncertainty of Moving-Object Representations. Technical Report CC-99-2, Chorochronos, Athens, Greece, April 1999. Available from `http://www.dbnet.ntua.gr/~choros/TRs/99/2/report.ps.gz`, accessed on 05-Jun-2001.

[77] G. Piccinelli. Service Provision and Composition in Virtual Business Communities. Technical Report Number 84, Hewlett Packard Extended Enterprise Laboratory, Bristol, England, July 1999. Available from `http://www.hpl.hp.com/techreports/1999/HPL-1999-84.html`, accessed on 22-Aug-2000.

[78] G. Piccinelli and L. Mokrushin. Dynamic Service Aggregation in Electronic Marketplaces. Technical Report Number 31, Hewlett Packard Laboratories, Bristol, England, February 2001. Available from `http://www.hpl.hp.com/techreports/2001/HPL-2001-31.html`, accessed on 04-Apr-2001.

[79] D. Plummer. An Ethernet Address Resolution Protocol, 1982. Available from `http://tools.ietf.org/html/rfc826`, accessed on 09-Oct-2006.

[80] Princeton University. WordNet - a lexical database for the English language. `http://www.cogsci.princeton.edu/cgi-bin/webwn` (17/01/2005).

[81] Rhizomik. ReDeFer - XSD2OWL, 2006. Available from `http://rhizomik.net/redefer/`, accessed on 15-Mar-2006.

[82] D. Roman, U. Keller, H. Lausen, J. d. Bruijn, R. Lara, M. Stollberg, A. Polleres, C. Feier, C. Bussler, and D. Fensel. Web Service Modeling Ontology. *Applied Ontology*, 1(1):77 – 106, 2005.

[83] N. S. Rosa, P. R. Cunha, and G. R. Justo. Process NFL: A language for describing non-functional properties. In *Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS'02)*, volume 9, page 282b, Hawaii, U.S.A, 2002.

[84] RosettaNet. RosettaNet Business Dictionary v2.1, 31st October 2002 2002. Available from `http://www.rosettanet.org/`, accessed on 14-Mar-2006.

[85] RosettaNet. RosettaNet, 2006. Available from `http://www.rosettanet.org/`, accessed on 14-Mar-2006.

[86] R. T. Rust and R. L. Oliver, editors. *Service Quality: New Directions in Theory and Practice*. SAGE Publications, Thousand Oaks, California, USA, 1994.

[87] A. Sahai, V. Machiraju, and K. Wurster. Managing Next Generation E-Services. Technical Report HPL-2000-120, Hewlett-Packard Laboratories, Palo Alto, California, USA, September 2000. Available from `http://www.hpl.hp.com/techreports/2000/HPL-2000-120.html`, accessed on 27-Dec-2000.

[88] S. Saltenis, C. S. Jensen, S. T. Leutenegger, and M. A. Lopez. Indexing the Positions of Continuously Moving Objects. Technical Report TR - 44, TimeCenter, Denmark, November 1999. Available from `http://www.cs.auc.dk/research/DP/tdb/TimeCenter/TimeCenterPublications/%TR-44.ps.gz`, accessed on 05-Jun-2001.

[89] T. W. Sandholm and V. R. Lesser. Advantages of a Leveled Commitment Contracting Protocol. In *Proceedings of the 13th National Conference on Artificial Intelligence and 8th Innovative Applications of Artificial Intelligence Conference (AAAI'96 and IAAI'96)*, volume 1, pages 126–133, Portland, Oregon, USA, 1996. AAAI Press.

[90] H. Schuster, D. Georgakopoulos, A. Cichocki, and D. Baker. Modeling and composing service-based and reference process-based multi-enterprise processes. In B. Wangler and L. Bergman, editors, *Proceedings of the 12th International Conference on Advanced Information Systems Engineering (CAiSE'00)*, Lecture Notes in Computer Science, pages 247–263, Stockholm, Sweden, 2000. Springer-Verlag.

[91] ServicesWeb.org. ServicesWeb.org, 2006. Available from `http://www.servicesweb.org/`, accessed on 27-Jun-2006.

[92] A. ShaikhAli, O. F. Rana, R. Al-Ali, and D. W. Walker. UDDIe: An Extended Registry for Web Services. In *Proceeduings of the Symposium on Applications and the Internet Workshops (SAINT 2003 Workshops) - Service Oriented Computing:Models, Architectures and Applications Workshop*, pages 85–89, Florida, MI, USA, 2003. IEEE Computer Society.

[93] S. Singh. Quality of Service Guarantees in Mobile Computing. *Computer Communications*, 19(4):359–371, 1996.

[94] Society for Worldwide Interbank Financial Telecommunications (SWIFT). Society for Worldwide Interbank Financial Telecommunications Web Site, 2001. Available from `http://www.swift.com/`, accessed on 08-May-2001.

[95] Standards Australia. Code of Tendering - AS 4120, 1994.

[96] L. S. D. I. Systems and U. of Georgia. Radiant: WSDL-S Annotation Tool, 2006. Available from `http://lsdis.cs.uga.edu/projects/meteor-s/downloads/index.php?page=1`, accessed on 15-Mar-2006.

[97] TC37/SC2-TC46/SC4 Joint Working Group. ISO 639-2, Codes for the representation of names of languages–Part 2: Alpha-3 code, 1998. Available from `http://www.loc.gov/standards/iso639-2/`, accessed on 11-Aug-2004.

[98] Tenderlink.com Pty Ltd. Tenderlink, 2005. Available from `http://www.tenderlink.com/`, accessed on 01-Aug-2005.

[99] A. H. M. ter Hofstede. *Information Modelling in Data Intensive Domains*. Doctor of Philosophy, University of Nijmegen, Nijmegen, The Netherlands, 1993.

[100] Y. Theodoridis, T. K. Sellis, A. Papadopoulos, and Y. Manolopoulos. Specifications for Efficient Indexing in Spatiotemporal Databases. In M. Rafanelli and M. Jarke, editors, *Proceedings of the 10th International Conference on Scientific and Statistic Database Management (SSDBM'98)*, pages 123–132, Capri, Italy, 1998. IEEE Computer Society.

[101] S. Thorsen. Time zone abbreviations, 2004. Available from `http://www.timeanddate.com/library/abbreviations/timezones/`, accessed on 08-Jul-2004.

[102] I. Toma and D. Foxvog. Non-Functional Properties in Web Services - WSMO Working Draft 16th June 2006, 2006. Available from `http://www.wsmo.org/TR/d28/d28.4/v0.1/20060616/d28.4v0.1_20060616.%pdf`, accessed on 21-Jun-2006.

[103] D. Trastour, C. Bartolini, and J. Gonzalez-Castillo. A Semantic Web Approach to Service Description for Matchmaking of Services. Technical Report Number 183, Hewlett Packard Labs, Bristol, England, July 2001. Available from `http://www.hpl.hp.com/techreports/2001/HPL-2001-183.html`, accessed on 22-Aug-2001.

[104] UDDI.org. UDDI Version 3.0.2 API Specification, July 2004. Available from `http://uddi.org/pubs/uddi-v3.0.2-published-20041019.pdf`, accessed on 17-Jan-2005.

[105] United Nations. United Nations Standard Products and Services Codes (UN/SPSC), 2000. Available from `http://www.unspsc.org/`, accessed on 15-Nov-2000.

[106] United Nations Centre for Trade Facilitation and Electronic Business. Core Components Technical Specification Part 8 of the ebXML Framework v2.01,

2003. Available from http://www.unece.org/cefact/ebxml/CCTS_V2-01_Final.pdf, accessed on 20-Jan-2006.

[107] United Nations Centre for Trade Facilitation and Electronic Business. UN/CEFACT Modeling Methodology (UMM) User Guide, 2003. Available from http://www.unece.org/cefact/umm/UMM_userguide_V20030922.pdf, accessed on 20-Jan-2006.

[108] United Nations Centre for Trade Facilitation and Electronic Business. Electronic Business Extensible Markup Language (ebXML) Part 5: ebXML Core Components Technical Specification, Version 2.01(ebCCTS), 2005. Available from http://www.iso.ch/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=4%1022&ICS1=35&ICS2=40&ICS3=&scopelist=.

[109] United States Patent and Trademark Office. General Information Concerning Patents, 2004. Available from http://www.uspto.gov/web/offices/doc/general/index.htm, accessed on 16-Nov-2004.

[110] United States Postal Service. Postal Addressing Standards: Publication 28, November 2000 2000. Available from http://pe.usps.gov/cpim/ftp/pubs/Pub28/pub28.pdf, accessed on 08-Jul-2004.

[111] University of Leipzig. XML2OWL Demonstration Platform, 2006. Available from http://xml2owl.sourceforge.net/index.php, accessed on 15-Mar-2006.

[112] W. M. P. van der Aalst and A. H. M. ter Hofstede. YAWL: yet another workflow language. *Information Systems Journal*, 30(4):245 – 275, 2005.

[113] W3C Web Services Team. Position Papers for the World Wide Web Consortium (W3C) Workshop on Web Services. Technical Number HPL-2001-73, Hewlett Packard Laboratories, Palo Alto, California, USA, April 2001. Available from http://www.hpl.hp.com/techreports/2001/HPL-2001-73.html, accessed on 21-May-2001.

[114] S. Weibel, J. Kunze, C. Lagoze, and M. Wolf. RFC2413 - Dublin Core Metadata for Resource Discovery, 1998. Available from http://rfc2413.x42.com/, accessed on 21-Nov-2004.

[115] Wikipedia. RFID, 2004. Available from http://en.wikipedia.org/wiki/RFID, accessed on 16-Nov-2004.

[116] World Intellectual Property Organization (WIPO). *The International Nice Classification of Goods and Services for the Purposes of the Registration of Marks*. World Intellectual Property Organization, 1977.

[117] World Intellectual Property Organization (WIPO). *Establishing an International Classification of the Figurative Elements of Marks*. World Intellectual Property Organization, 1985.

[118] World Intellectual Property Organization (WIPO). *Intellectual Patent Classification (IPC) Complete Set*. World Intellectual Property Organization, 7 edition, 1999.

[119] World Intellectual Property Organization (WIPO). *International Classification for Industrial Designs (Locarno Classification)*. World Intellectual Property Organization, 8 edition, 2003.

[120] xcentric technology and consulting GmbH. JAXFront/XUIEditor Version 2.0, 2006. Available from `http://www.jaxfront.com/`, accessed on 21-Apr-2006.

[121] YellowPages.com.au. Yellow Pages Australia, 2006. Available from `http://www.yellowpages.com.au/`, accessed on 14-Jun-2006.

[122] G. Zacharia, A. Moukas, and P. Maes. Collaborative Reputation Mechanisms in Electronic Marketplaces. *Decision Support Systems*, 29(4):371–388, 2000.

[123] V. A. Zeithaml and M. J. Bitner. *Services Marketing*. McGraw Hill, New York, New York, USA, 1996.

[124] V. A. Zeithaml, A. Parasuraman, and A. Malhotra. A Conceptual Framework for Understanding e-Service Quality: Implications for Future Research and Managerial Practice. Research Report 00-04, Kenan Flagler Business School, University of North Carolina, Chapel Hill, North Carolina, 2001. Available from `http://www.kenanflagler.unc.edu/Marketing/wpapers/papers/e-Service.pdf`, accessed on 10-Jul-2001.