QUALITY OF SERVICE SUPPORT FOR SERVICE DISCOVERY AND SELECTION IN SERVICE ORIENTED COMPUTING ENVIRONMENT

Vikas Deora

School of Computer Science
Cardiff University

This thesis is submitted in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

· December 2007 ·

UMI Number: U585091

UMI

Dissertation Publishing

UMI U585091

ProQuest

Abstract

Service oriented computing (SOC) represents a new generation of web architecture. Central to SOC is the notion of *services*, which are self-contained, self-describing, modular applications that can be published, located, and invoked across the Internet. The services represent capability, which can be anything from simple operations to complicated business processes. This new architecture offers great potential for e-commerce applications, where software agents can automatically *find* and *select* the services that best serve a consumer's interests. Many techniques have been proposed for discovery and selection of services, most of which have been constructed without a formal Quality of Service (QoS) model or much regard to understanding the needs of consumers. This thesis aims to provide QoS support for the entire SOC life cycle, namely: (i) extend current approaches to service discovery that allow service providers to advertise their services in a format that supports quality specifications, and allows service consumers to request services by stating required quality levels. (ii) support matchmaking between advertised and requested services based on functional as well as quality requirements. (iii) perform QoS assessment to support consumers in service selection. Many techniques exists for performing QoS assessment, most of which are based on collecting quality ratings from the users of a service. This thesis argues that collecting quality ratings alone from the users is not sufficient for deriving a reliable and accurate quality measure for a service. This is because different users often have different expectations and judgements on the quality of a service and their ratings tend to be closely related to these expectations, i.e., how their expectations are met. The thesis proposes a new model for QoS assessment, based on user expectations that collects expectations as well as ratings from the users of a service, then calculates the QoS using only the ratings which were judged on similar expectations.

Contents

List of Figures

List of Tables

Citations to Previously Published Work

Large portions of Chapters 4, 5 and 6, as well as parts of Sections 3.2 have been presented in the proceedings of peer reviewed international conferences and journals:

"Modelling Quality of Service in Service Oriented Computing", V.Deora, J.Shao, W.A.Gray, and N.J.Fiddian, In Proceedings of IEEE International Symposium on Service-Oriented System Engineering, 2006, pages 95-101.

"Supporting QoS Based Selection in Service Oriented Architecture", V.Deora, J.Shao, W.A.Gray, and N.J.Fiddian, In Proceedings of International Conference on Next Generation Web Services Practices, 2006, pages 117-126.

"Expectation-based quality of service assessment", V.Deora, J.Shao, W.A.Gray, and N.J.Fiddian, International Journal on Digital Libraries, March 2006, Volume 6, Issue 3, pages 260-269, Springer Berlin.

"Supporting formation and operation of virtual organisations in a grid environment", J.Shao, W.A.Gray, N.J.Fiddian, V.Deora, G.Shercliff, P.J.Stockreisser, T.J.Norman, A.Preece, P.Gray, S.Chalmers, N.Oren, N.R.Jennings, M.Luck, V.D.Dang, T.D.Nguyen, J.Patel, L.Teacy and S.Thompson, In Proceedings of UK e-Science All Hands Meeting, 2004, ISBN 1-904425-21-6.

"Incorporating QoS Specifications in Service Discovery", V.Deora, J.Shao, G.Shercliff, P.J.Stockreisser, W.A.Gray, and N.J.Fiddian, In Proceedings of Second International Web Services Quality Workshop, 2004, pages 252-263.

"Agent-based Formation of Virtual Organisations",T.J. Norman, A.Preece, S.Chalmers, N. R.Jennings, M.Luck, V. D.Dang, T. D.Nguyen, V.Deora, J.Shao, W. A.Gray, and N.J.Fiddian, In International Journal of Knowledge Based Systems, 2004, Volume 17, Issues 2-4, pages 103-111.

"A Quality of Service Management Framework Based on User Expectations",V.Deora, J.Shao, W.A.Gray, and N.J.Fiddian, In Proceedings of International Conference on Service-Oriented Computing, 2003, pages 104-114.

"CONOISE: Agent-based Formation of Virtual Organisations", T.J. Norman, A. Preece, S. Chalmers, N. R. Jennings, M. Luck, V. D. Dang, T. D. Nguyen, V. Deora, J. Shao, W. A. Gray, and N. J. Fiddian, In Proceedings of 23rd SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence, 2003, pages 353-366.

Acknowledgments

I would like to first-most thank the person without whom I would have never achieved this thesis. He has not just contributed towards teaching me the meaning and importance of research but brought about changes in the way I think, which has made me a different person from when I started my research. This person is my mentor, my supervisor Dr. Jianhua Shao. Dr. Shao, has been a figure of motivation, a great listener and been a pillar of support throughout the period of my PhD. He always pushed me to bring the best out of me, always had confidence in me and worked very hard with me to teach me how to investigate and identify and tackle real research problems. I will always be in his debt, for sharing his wisdom and knowledge with me, without which I would have never reached this stage in my life.

I would also like to thank my supervisor Professor Alex Gray. He has been the heart and soul of my research work. He provided the opportunity for me to do the research, without his support, I would have never started this research. He has looked after me and made sure that I stay away from troubles. I want to thank him for taking so much time to support and guide me - despite his very busy schedule. Without his guidance and leadership I would not have been able to complete the thesis.

I thank Professor Nick Fiddian for being supportive throughout my PhD process. He provided valuable feedback during write-up of my research publications and supported me to attend many important workshops and conferences in my research area.

My thanks also goes to Professor Omer Rana. He provided me with the job opportunity as a research assistant so I can continue to work on my research and finish writing-up without facing financial difficulties. He has been very understanding

and supportive during the whole process.

Furthermore, I would like to thank Mrs. Margaret Evans for all the administrative support throughout the research process, including finding appointment dates for me to see Professor Alex Gray. I would also like to thanks Mrs. Helen Williams who on many occasion helped me with my administrative problems and also in completion of my thesis. Her constant effort into monitoring the progress of research students allowed me to build up many written progress report that I was able to refer to when writing this thesis.

I would like to thank my life partner Shrija Rajbhandari, she helped me every step of the way in this research journey. From reading my thesis, being critical of my work, encouraging me, motivating me and most importantly keeping me focussed on my work. She made lots of sacrifices and juggled many work and even took care of me when she was not at her best. Without her nothing would have ever been possible.

I am most grateful to my parents for their love, blessings and affection. Even though they did not completely understand what I was doing, they provided me with the moral support that was instrumental to the completion of my thesis. They worked very hard so that I can have a good education and sacrificed a lot, especially in not having me around for years. I dedicate this achievement to them for all their support.

Chapter 1

Introduction

The development and use of technology can always change how people interact with each other and consume services available to them. The wide-spread use of World Wide Web has led to one of the biggest changes in how people select and use services. For example, getting the latest music or movies, making phone calls to friends, and getting the latest world news today are much different from 20 years ago. To exploit this potentially huge market, businesses will need to make some fundamental changes in the way services are advertised and made available to consumers.

There is a growing interest in Service oriented computing (SOC) in recent years [35, 106, 150, 159]. Central to SOC is the notion of a *service* which can broadly be considered as a software component that represents some computational or business capability. By allowing services to be advertised declaratively, discovered dynamically and invoked remotely, SOC makes it possible for users to locate, select and execute services without having to know how and where they are implemented. This new computing paradigm offers great potential for e-commerce applications [74, 82, 135].

1

For example, vendors may wish to identify suitable partners from time to time to form a virtual organisation [133, 148], so that they together can compete better in the market, and consumers would always want to select the services that best serve their interests. All such "matchmaking" tasks can potentially be performed by software agents in an SOC environment automatically.

In order to achieve the full potential that SOC can bring to the next generation of e-commerce, some challenges must be addressed.

1. Service Discovery: Existing techniques and standards [97, 9] are designed to provide a framework for describing services in terms of what they can do and how they may be invoked. The searching for required services is conducted based on such specifications. This is inadequate because it does not include support for the non-functional aspects of a service, for example, Quality of Service (QoS). Non-functional aspects of a service are important when considering service discovery, because, just like in any other marketplace, it is possible to have several service providers in an SOC environment, who offer broadly similar services. This means QoS is often an important criteria by which service providers can distinguish themselves from other providers, and service consumers can select the most appropriate service. It is essential, therefore, that services can be selected based not only on the required functionality, but also QoS requirements.

   To incorporate QoS into service discovery, it is essential that existing techniques are extended to allow service providers to describe QoS for their services. This will then allow service consumers to locate a service that may offer the level of quality they require, and service providers to distinguish themselves in the

market. Since in the real world, consumers are driven rationally to strike a best possible deal, service discovery with such a QoS augment will mirror the real world and function better.

2. Service Selection: Since service providers will be competing for consumers by offering similar services and advertising similar QoS levels, which service to select becomes an important issue. Since a provider may not always provide what they promise in an advertisement, a consumer must individually assess a service based on not just what is advertised by the service provider but also on its past performance.

In this thesis, we address these two important research challenges which are associated with using SOC in an e-commerce environment. We extend existing service discovery models by including support for QoS specifications. This is achieved by allowing service providers to describe services with QoS specifications which consumers use to discover services with their QoS requirements. We also augment service selection by allowing QoS assessment to be performed on a service. The aim of this assessment is to rank services in terms of their QoS, which will then assist the consumer in making an appropriate service selection decision.

1.1 Research Aims

The preceding discussion illustrates that SOC provide an appropriate infrastructure for performing automated service discovery and selection for consumers in service-based environments, like e-commerce. It should also be apparent that such environ-

ments may potentially include service providers delivering different levels of quality. As such, issues of QoS naturally arise from these, and consumers need to be aware of them so that they can make effective decisions on which providor(s) to use. It is this need that forms the motivation of this thesis and the basis of our work. In particular, we aim to *develop QoS augmented discovery and assessment models, or mechanisms, that can be used to aid decision making by consumers in a service oriented environment.* In doing so, we can allow consumers to select services in a service-based environment effectively, by identifying which services are most reliable, and which are best avoided based on their past performances. This aim raises four goals:

- The study of existing QoS approaches in various application domains.

- The application of QoS approaches to service discovery and selection in an SOC environment.

- The investigation and design of QoS annotated service capability description and analysis of such QoS assertions for service discovery.

- The investigation of factors that should be accounted for, when designing QoS-based service assessment mechanism in an SOC environment.

The first general purpose of this thesis consists of background research into existing QoS approaches in different domains, which will provide us with a general view of quality. To address the second goal, we need to identify and analyse the benefits and use of the QoS approaches in order to evaluate their applicability to service discovery and selection in an SOC environment. To address the third goal, we need to consider how QoS annotation in service description is used in finding providers willing to

provide a particular service. Finally, to address the fourth goal, it is essential to consider the information that is available to service assessment mechanism involving QoS. This very much depends on the situation, but examples include using consumer expectations and observations of past service behaviour. This directs us to the main research objectives as follows:

- To design a QoS augmented model that incorporates: (1) a service discovery mechanism, and (2) service assessment in an SOC environment.

- Modelling a structured representation of QoS for advertising and discovery of services in an SOC environment.

- To design a matchmaking mechanism to allow consumers discover appropriate service advertisements.

- Investigate methods to collect and analyse the QoS information to be used in the service assessment for selection purpose.

- Perform evaluations to assess the reliability of service assessment mechanisms for selecting potential service providers.

Together, these objectives constitute a set of requirements that a QoS augmented model must achieve, if it is to facilitate effective service selection decisions. As should become clear from Chapter 2, each of these requirements is fulfilled, at least in part, by mechanisms described in the existing literature. However, these existing approaches all suffer from certain limitations, which are outlined in Section 1.2 and addressed in this thesis. Specifically, we make a number of contributions to the state of the art, which are outlined in Section 1.3.

1.2   Research Problems and Approach

Various methods for modelling, calculating and monitoring QoS have been proposed in the literature [12, 36, 103, 139], especially for web services and multi-agent systems [187, 208, 215]. However, all these approaches suffer from limitations in their definitions of quality.

1. Service Discovery: Existing work on QoS augmented service discovery is largely associated with network quality parameters, such as bandwidth and jitter, or are limited to a particular domain of interest [12, 36]. In e-commerce, however, QoS involves much more than just network attributes. While these parameters vary from application to application and can change as new domain requirements are introduced, they are still limited. To overcome the current limited QoS representation and allow more dynamic service discovery, an approach that can provide an extensible QoS description is needed, which will allow new QoS requirements to be used as and when they become required and their representation is determined.

   In this thesis, we propose to extend existing standard service description schemas to incorporate QoS information. Some QoS attributes, for example, availability, response time and reliability, are general enough to apply to all application domains, whereas other attributes, such as frame-rate and update frequency, are domain specific. Thus our aim is to create a QoS schema designed to allow representation of domain independent QoS attributes as well as those that are domain specific.

2. Service Selection: Current approaches to supporting service selection have been largely designed to establish trust and reputation and use them in the selection process [127, 84, 208]. They collect quality ratings from the users of a service and then aggregate them in some way to derive a quality verdict for the service. However this approach suffers from two fundamental weaknesses: (1) consumers are invited to rate a service in "absolute" terms, e.g., "Good", "Average", "Bad", or "3 out of 5". Such quality ratings may not be very meaningful or can even be misleading in some cases, because the context within which the ratings are derived is not known, e.g., the reasons for a consumer to rate a service "Good". (2) the aggregate quality rating for a service is derived "statically" using all the ratings collected from the users. This does not take into account the fact that some of the ratings may not be relevant to a particular quality assessment request. For example, if a consumer expected a service to be exceptionally good and only found it to be average, such a quality rating may not be useful to a QoS assessment for a consumer with an average expectation for the service.

In this thesis, we propose to address these two problems by introducing a new model that collects and calculates QoS "relatively". That is, we attempt to collect from service users QoS ratings as well as their expectations on QoS, so that we can measure QoS in relative terms, i.e., how well a delivered service meets users' expectations. Based on user expectations, we also propose to calculate the quality of a service dynamically at the time a request for QoS assessment is made, and use only the ratings of assessors that have similar

expectations as this should give a more focussed measurement.

1.3   Research Contributions

The main focus of this thesis is to develop support for enhanced service discovery
and selection by allowing QoS attributes of a service to be specified, searched and
evaluated. This is achieved by (a) searching for services that claim to offer the required
QoS, and (b) assessing how well each service can actually provide the promised QoS
based on user expectations. The thesis makes the following main contributions.

- A generic framework for supporting QoS augmented service discovery and se-
  lection in an SOC environment.

- A new service discovery approach that introduces a QoS schema to allow service
  providers and service consumers to describe QoS offers and requirements, respec-
  tively. The QoS schema introduced is extendable to allow a new QoS vocabulary
  to be described. The proposed discovery approach provides a matchmaking al-
  gorithm to allow service consumers to find the most appropriate services based
  on QoS offerings.

- A service selection approach that provides QoS assessment rankings on ser-
  vices. The proposed selection approach captures QoS ratings as well as con-
  sumer expectations for services. This allows us to measure how well a delivered
  service meets a consumer's expectations. Our experiments show that the pro-
  posed approach enables a personalized QoS assessment and can result in a more
  meaningful selection of service which is closer to a consumer's expectations.

1.4 Thesis Structure

Chapter 2 reviews the current state of the art in service discovery and selection. We describe some of the most influential approaches to service discovery that are competing to become future standards for SOC. We discuss different service description efforts that have attempted to incorporate QoS in service discovery. We then discuss service selection and assess related approaches that are most relevant to performing QoS assessment.

Chapter 3 describes the design of a general framework for supporting QoS enhanced service discovery and selection in an SOC environment. The proposed framework is illustrated using a Virtual Organization (VO) formation scenario and a multimedia use case. This chapter also presents the high level interactions between the architectural components for discovering services, and collecting and assessing QoS.

Chapter 4 extends the current approaches to service discovery by allowing service providers and consumers to express their promises and requirements for QoS. We propose a model for incorporating QoS specifications and requirements in service discovery, and describe how matchmaking between advertised and requested services based on functional as well as QoS requirements can be achieved.

Chapter 5 argues that collecting quality ratings alone from the users is not sufficient for deriving a reliable or accurate quality measure for a service. This is because different users often have different expectations for QoS and their ratings tend to be closely related to their expectations, i.e., how their expectations are met. We propose

a QoS assessment model based on user expectations. That is, we collect expectations as well as ratings from the users of a service, then calculate QoS using only the ratings made by users with similar expectations.

Chapter 6 describes a simulated test environment that allows various properties and methods used by QoS assessment techniques to be studied. Also included is a comparison of the proposed approach and the common QoS assessment technique that averages all ratings. Our experiments show that the method we propose can result in more meaningful and reliable quality ratings for services.

Chapter 7 concludes the thesis by providing a summary of contributions made by the research presented in the thesis. Also we discuss some future work in this chapter.

Chapter 2

Related Work

This thesis aims to address issues arising in service discovery and selection when QoS augmentation is being used. There is a detailed discussion of these issues and the related research. First, we provide a general overview of some of the fundamental research involved in developing technologies for service discovery and review some of the major developing technologies that support QoS specification. The analysis of relevance and weakness of these major technologies is also presented. Secondly, a review of literature is provided in different application areas where QoS assessment is used as part of service selection. Different systems applicable to service selection are evaluated, with the aim to understanding the characteristics of their operational models.

2.1 Service Discovery

Booth el al. [24] define service discovery as *the act of locating a machine-processable description of a Web service-related resource that may have been previously unknown and that meets certain functional criteria. It involves matching a set of functional and other criteria with a set of resource descriptions.* We adopt this definition of service discovery for rest of the thesis.

Service discovery in current applications [63, 158] is still largely a process involving the use of search engines [90, 112] and performing text matching [219] based on keywords to locate required services. This process results in the problem of poor precision (a large number of irrelevant services are retrieved) and bad recall (a large number of relevant services are not retrieved) [203, 13]. This is due to the fact that HTML does not provide adequate support for specifying a service's exact capabilities. Nevertheless, in the past decade, research into automated service discovery mechanisms [78, 186, 132, 10] has gained a momentum and allows capability based service description (advertisements), representation of service requests, and matching service requests with service descriptions.

The work presented in [58] was one of the first research efforts that introduced the concept of capability description and used facilitator agents to select the best possible service based on matching the service request and the described capabilities of the service provider. A series of work after that were performed in the area of matchmaking and service discovery, both in the agent community [190, 217, 152, 94] and more recently in SOC [108, 71]. These approaches provide a platform for service providers to advertise their services and service consumers to discover and

invoke the services. However, such service discovery approaches do not consider QoS requirements, which are important for finding services that actually provide required QoS levels [98]. In an e-commerce environment where a large number of competing services are available, it is imperative for a service provider to differentiate its services from others based on the QoS guarantees it can provide.

So far, little research has been done to integrate QoS into service discovery [38, 151, 205]. Progress on supporting dynamic service discovery in SOC is largely represented by the development of UDDI [3], which is currently a *de facto* standard for service advertisement and discovery. UDDI does not offer any support for QoS specification and relies largely on keyword matching for service discovery. Recently, some extensions have been made to UDDI to allow inclusion of QoS in service specifications [38], but they support only a very limited set of specific attributes, for example, network response time. One of the disadvantages of these earlier attempts to use QoS augemented service discovery [166], is that they were unable to define QoS classes over a wide range of QoS attributes to provide richer semantics for describing services to be specified and more sophisticated matchmaking between advertisements and service requests to be performed.

Various service description languages, such as WSDL [4], DAML-S [9], OWL-S [2] and ebXML [1] have been proposed and are competing to become future standards for describing services. The Web services registered in UDDI are described in WSDL [192], and together they provide a framework for service description and discovery. This mechanism only supports description of basic information about services, such as service name and methods to invoke the services. Integrating semantic description

of services into UDDI may enhance the discovery process by allowing to describe and use functional and non-functional aspects of services. DAML-S, and its newer version OWL-S, were developed by the semantic web community for describing services at a generic level. They support the use of a service ontology for service description and allow some form of reasoning in service discovery [50, 51]. Although these languages are quite powerful in describing the functional aspects of services, they offer little support [166, 218] for specifying QoS for service discovery within a SOC environment. More recently, attempts [21, 218] have also been made to incorporate support for QoS specification using DAML-S. However, the proposed approaches to modeling QoS tend to be based on a few domain specific attributes and are not extensible to cater for other domains and new attributes. The exception is the work reported by Liu [113], which attempted to create a QoS model that is extensible. Liu proposed a QoS computation model for Web service discovery by implementing a QoS registry and demonstrated this model using a hypothetical phone service application. This approach has the following characteristics. First, new QoS attributes can be added to a service description without affecting the underlying QoS discovery algorithm. Second, user preferences are used to create a list of discovered services ordered by QoS rankings. Third, it allows a fair QoS data collection mechanism that includes QoS score from not just the service providers but also from monitoring services and users' feedback. This work concentrated on the penalty and compensation aspects of QoS management, and lacks support for representing consumers' QoS requirements. As such, it is not useful for our service discovery purpose.

Some approaches to QoS enhanced service discovery have also put their efforts

into identifying important QoS attributes [124, 29, 195]. For example, Mecella et al. [124] have designed a broker which can select the best available data from a number of different service providers, based on a specified set of data quality dimensions, e.g., accuracy, completeness, currency and consistency. Work presented by Burgess et al. [29] has presented a QoS taxonomy that defines an extensive set of QoS classes and attributes that can be easily customized for most information retrieval systems. These studies, however, aim to identify and establish suitable quality metrics for services in specific application areas, so that the quality of these services can be meaningfully gauged. However, they do not aim to define how QoS requirements and capabilities can be represented and service discovery supported in an SOC environment.

Integrating QoS with service description has also been considered in work on deriving service level agreements (SLA) [100, 185, 171, 57, 69] for Web services. SLAs are often custom-made and negotiated between service requesters and service providers. The aim of such approaches is to incorporate QoS into service descriptions, so that certain service performance guarantees can be agreed between the provider and consumer. It is assumed that a consumer has already discovered a service provider for the required service and the task is to establish a contract. Considering QoS during discovery would enable retrieval of services that meet the user's QoS requirements and this can then help negotiation in SLA formation.

In the rest of this section we provide an overview of some of the approaches to QoS specification.

2.1.1 Web Service Level Agreement (WSLA)

IBM developed Web Service-Level Agreement (WSLA) [114] which uses an XML Schema [25] to define the language constructs of the WSLA framework. WSLA provides the necessary framework for service providers and consumers to define a contract of agreed service usage between them. The WSLA framework provides this by allowing service providers to specify their offers, i.e., the service or resource they are willing to provide, as service level objectives, action guarantees and metrics. The service level objectives state the service guarantee agreed by the provider, the action guarantees cover measures to be taken when a guarantee is not met and metrics specify how each service parameter is to be measured and how aggregated measures may be derived. Together these WSLA components provide functionality for a service provider to describe the agreed service provision and allow resource monitoring or accounting components to be traced if the agreed service level is not met. WSLA is a widely accepted and popular semantic framework for monitoring and management of SLAs once service providers are identified by a service consumer and contracts have drawn up [169, 134, 115, 44]. WSLA forms an important complimentary tool alongside any service discovery mechanism for enabling monitoring of services usage using SLAs and ensuring agreed service levels are delivered.

While WSLA solves the problem of service description, it lacks provision of (i) QoS semantics: WSLA aims to describe the functional details of service and acknowledges that nonfunctional aspects such as QoS still need to be developed, and (ii) matchmaking facilities between service consumer and provider: WSLA also lacks a capability for service providers to advertise their QoS and for service consumers to

request services by specifying QoS requirements.

2.1.2 UDDIe

UDDIe [166] extends the existing standard established by the UDDI [43] working group. UDDIe supports QoS discovery in the context of Grid Computing and allows service consumers to choose service providers based on QoS attributes such as bandwidth, CPU, and memory requirements. The implemented framework follows a broker architecture where consumers can send their requests for services with QoS properties to the QoS broker and the broker interacts with the UDDIe registry to fulfill the request. The broker functionality is implemented as part of the G-QoSM framework [6]. The broker is not part of the UDDIe registry, but utilises it to record and query about services with specific QoS attributes. The UDDIe registry performs a string search to check functional requirements sent by the broker against the stored service descriptions and formulates a reply with the resultant services that meet the requirements. The broker selects services based on the service consumer's QoS preference using a *weighted average* selection algorithm.

UDDIe supports service discovery within the context of Grid computing, but it suffers from numerous problems. First, it provides support for only one QoS parameter (i.e., bandwidth trading). It is essential to include domain independent QoS parameters, as well as any domain specific ones for discovery. Second, the design UDDIe is tied to a specific WSDL release. This makes maintenance difficult as all development effort will need to be reiterated for future WSDL releases. Finally, it lacks support for semantic service description [9], i.e. UDDIe does not allow matchmaking

other than exact equality to be performed during service discovery. This means the system can not enhance a basic user query to overcome problems due to spelling, synonymy and other common causes of error.

2.1.3 QML

HPs QoS modeling language (QML) [56] offers three abstract classes for service and QoS specification: contract type, contracts, and service profiles. A contract type defines QoS attributes with range and type of values that can be attached to each QoS attribute, for example, 99 percent availability. A contract is an instance of a contract type and thus represents an instance of a particular QoS attribute (e.g., ABC news service availability). A service profile associates contracts with a service interface and operations describing a service's functionality. QoS contracts can be specified at the service level, e.g., response time < 2ms, or at the operation level, e.g., update frequency > 10 per day.

Benefits of using QML include the following. First, QML supports generic QoS specification that support a variety of services and allows any QoS attributes from different application domains to be specified. Second, QML provides an easily extensible framework through their approach of contract and profile "refinement". A refinement statically defines a relationship between two contracts or service profiles. For example, it is possible to create a new contract N based on another contract B by specifying additional QoS attributes not present in B or by replacing QoS values in B. The QML project targets mapping QML to Java, so that Java objects represent QML specifications at runtime. However, QML's mapping process and matchmaker

are not yet clearly defined, to the best of our knowledge.

2.1.4 QoSME

One of the earliest approaches to creating a QoS specification language for distributed applications was carried out in the QoSME project [194]. The QoS Management Environment (QoSME) uses the Quality-of-Service Assurance Language (QuAL) to enable the specification of how to allocate, monitor, analyze and adapt to delivered QoS. QuAL enables applications to express their QoS needs and QoSME automatically generates the instrumentation to monitor the QoS of the applications. The QoSME collects statistics on the delivered QoS, by analyzing interactions among the applications, communication protocols, and operating systems (OS) and stores these statistics in a repository. QuAL also allows applications to express how to handle QoS violations. For example, distributed database applications may be very sensitive to data loss and may tolerate high transmission delays. Incorporating mechanisms to express the handling of QoS attributes such as 'transmission time' that are application aware, allows QuAL to create precise QoS descriptions.

QuAL supports specification of application level abstraction and system level objects. However, much of the support is provided for describing how to specify QoS requirements in terms of abstract representations, that are independent of the underlying network and OS. This allows a service consumers to specify requests for QoS in a language that is easy to understand, for example, using the frame rate instead of CPU capacity required and mapping this to underlying resource requirement when needed. QuAL also allows QoS attributes and constraints to be specified to support

negotiation between QoS offerings and the QoS requirements. The QuAL approach gives a rich and practical QoS specification and allows customized handling of QoS violations. This can help generate precise service offerings and requests to be used in service discovery. However, like many other languages introduced in last decade, QuAL lacks reusability: its constructs are designed to be part of the implemented software code, which makes it difficult to maintain.

2.1.5 WS-Policy

The WS-Policy [55] framework comes from a collaboration of large vendors (BAE, IBM, Microsoft and SAP). It is a general model and syntax to describe policies of a Web service. It is a flexible and extensible framework and a model that can be used for expressing capabilities, requirements and general characteristics of services in a Web service environment. Any specialized assertion (e.g., privacy policy and QoS characteristics) needs to be defined in addition to the generic policy. Thus, QoS requirements can be expressed as assertions in the WS-Policy model that can be associated with a Web service. WS-Policy is similar in pattern to DAML-S and OWL-S [9] which can only represent functional aspects of services. Both WS-Policy and DAML-S frameworks have had extensive support from the research community [177, 16, 174, 23] and will be equally influential in designing future standards for service description. Ludwig et al. [115] presented an extension to WS-Policy with QoS, which allows QoS terms to be taken from QoS vocabularies defined using XML. This work however is developed solely for QoS monitoring.

The WS-Policy benefits include its extensible and open framework, that allows

QoS extensions to be plugged-in by others and a single policy grammar to allow QoS extended assertions to be reasoned with in a consistent manner. Use of WS-Policy over other service description languages such as DAML-S causes architectural constrains caused by its requirement to use the Web services protocol stack, i.e., WSDL [40] (for describing services), UDDI [43] (repository to register services) and SOAP [68] (for communication protocol). WS-Policy does not specify how policies can be discovered within the Web service framework. Thus it is limited in its service discovery ability based on policy information such as QoS.

2.1.6 Summary of QoS description and discovery approaches

Five approaches to service description and discovery have been reviewed. A summary of the findings is presented in Table 2.1, which highlights the main features of each approach and analyses their relevance and weaknesses.

The main features for all the approaches were identified and compared based on the following criteria:

1. Expressiveness: Able to represent QoS for a variety of domains

2. Reusability: QoS representation reusable in other queries and ability to add other Service description specifications

3. Simplicity: Easy to understand and adopt

4. Extensibility: Easy to incorporate new QoS semantics

5. Flexibility: Easy to separate the semantics from the underlying implementations for maintenance.

	Application Domain	Main Features	Relevance to Service Discovery	Weakness in supporting Service Discovery
WSLA	Service Level Agreement	Expressive, Reusable, and Extensible	Consumer QoS requirements can be matched with a list of SLAs for service discovery.	SLAs do not necessarily represent providers capability and QoS augmented advertisements are more suitable.
UDDIe	Bandwidth Broker	Simple	Supports matchmaking of consumer QoS requirements for service discovery.	Only supports exact matching using service name or URI address, no enhancement of query.
QML	Service Level Agreement	Expressive, Simple, Extensible, and Flexible	Consumer QoS requirements can be matched with a list of contracts for service discovery.	Contracts only represent past commitments but not current QoS capabilities.
QoSME	Service Monitoring	Expressive, Extensible, and Flexible	Monitored QoS data about a service provider can be matched with consumer requirements for service discovery.	Restricts discovery over monitorable QoS attributes only. Does not represent providers current QoS capability.
WS-Policy	Privacy and Security	Reusable, Simple, and Flexible	Discovery of services with similar policy constraints on privacy and security can be performed.	Limited use to other aspects of QoS.

Table 2.1: QoS Specification and Service Discovery Approaches

As can be seen from Table 2.1, the approaches that are simple lack either expressiveness and extensibility or reusability. Also, some of the approaches that are expressive, extensible and flexible are not reusable. In terms of their relevance, most of the approaches provide support for QoS specification, but lack support for repre-

senting consumers requirements and service advertisements. Some of the approaches can be used to support service discovery, but are either limited in their QoS representation capability or restricted to a specific application domain. The approach presented in this thesis aims to provide QoS representation that attempts to satisfy the above criteria, with a focus to support service discovery.

2.2 Service Selection

Service discovery mechanisms enable services to be discovered based on functional and non-functional criteria, but they do not help select a "best" one among the multiple services that appear to meet the service consumer's requirements. One possible solution would be to select a service randomly or let the service consumer to choose one. Such an approach is, however, not effective, as there is no guarantee that the randomly selected service would provide acceptable QoS without further information. Some real world applications, such as EBay [53], Amazon [8] and MovieLens [67], attempt to collect from users information about their experience with the services they used and share this experience with other users to help their service selection. There also exists a large body of research work in the area of using trust and reputation [117, 127, 164, 84, 208, 121] to determine QoS. These approaches ascertain the level of trust that can be placed on an advertisement by evaluating past service provisions by the provider. The service trust is then calculated based on the information collected either directly through QoS monitoring techniques [121] or through social networks [127, 208] based on users that are willing to contribute information about their experience. However, these approaches are not adequate for service selection as

they do not involve personalized QoS assessments that are based on the consumer's expectation.

Work in the area of collaborative filtering [154, 75, 123, 85, 7, 18, 126] attempts to create customized recommendations for consumers, based on ratings from other users who share similar interests. These approaches however do not assess QoS. Work on QoS assessment is also well established in the area of business studies. However, no attempts have been made so far to study these business models [141, 92, 173, 168] for QoS assessment in supporting service selection in an SOC environment.

Table 2.2 summarizes seven major research projects that are relevant to service selection with their application areas, main features and applicability listed. The criteria for comparing main features of these systems were based on:

- *User QoS rating*: Utilization of user ratings in decision making.

- *Context behind rating*: Able to incorporate and use the reason for a user's ratings during service assessment.

- *Personalized assessment*: Able to assess a service that is personalized for a user.

The literature shows that, some systems perform personalized assessment for users without considering QoS; systems differ in terms of their service assessment model, i.e. use different QoS dimensions and context in determining the rating; and some systems are specific to a particular domain, such as workflow systems, and trust and reputation systems. In reviewing the literature we categorize service selection in terms of their research areas as follows:

- Reputation and Trust

	Research Area	**Main Features**	**Use and Benefits**
WSAP	Reputation and Trust	User QoS rating	Use QoS attributes and reputation for service selection
AgFlow	Workflow QoS	User QoS rating	Aggregates all monitored QoS data for service assessment and selection
GroupLens	Recommender System	Personalized assessment	Gives recommendations of services to a user from users that share similar rating patterns
SERVQUAL	Business Studies	User QoS rating, Context behind rating	Measures service quality by deducting users' expectations from their perceptions
CrossFlow	Workflow QoS	User QoS rating	Supports service selection based on time and cost QoS constraints
ReGreT	Reputation and Trust	User QoS rating, Context behind rating	Assesses reputation of a service and models the context behind the reputation
MovieLens	Recommender System	Personalized assessment	Recommendations of movies based on ratings from users who viewed similar movies

Table 2.2: Major Systems for Service Selection

- QoS in Workflow

- Recommender Systems

- Business Studies

In this section (2.3), we review the relevant service assessment techniques under

these four categories.

2.2.1 QoS, Reputation and Trust

There exists a large number of proposals in the literature for calculating QoS using reputation or trust of the service providers. These approaches, e.g., [117, 127, 164, 84, 208, 121], typically seek to establish the quality of a service by gathering ratings from the users who have used the service. Trust is defined as *the extent to which the service consumer is willing to depend on the service provider in a given situation, even though negative consequences are possible* (adapted from [83]). Trust assessment is normally considered to have a binary outcome [127, 84, 212], which is used by the consumer to determine the actions to be taken. In the area of service selection this action involves making a decision to choose a particular service for invocation. Reputation is considered different to trust and is defined as *the perception that a service provider creates through past actions about its intentions and norms* (adapted from [127]). Reputation systems consider assessment of a service's reputation from publicly available information only, e.g., user ratings of the service. Trust, on the other hand, implicitly assumes that the user may have some additional private information that he or she can use to overrule the general reputation given to a service. Many approaches to modelling trust [127, 208, 204, 70] are however based on reputation systems as they directly relate good a reputation score to a high trust value.

One of the earliest and most cited work on trust is presented by Marsh [120]. This work attempted to create a computational model of trust. His work is drawn from theoretical studies with a social and psychological foundation and thus is very

complex. This is not suited for use in an SOC environment for service selection as the work focuses only on service consumers who maintain private service experiences and do not consider sharing this information in a social network. Abdul-Rahman and Hailes [5] aimed to simply extend March's trust model. This extension is based on the presence of social networks in which user experiences are recorded and available throughout the network. Though a step in the right direction, this approach suffers from two flaws. First, it requires each user to maintain global knowledge about the whole community, which may be difficult to achieve. Second, trust is calculated as an average of all the ratings given to a service provider. This provides a general trust measure for a service provider, which may not be useful to a specific consumer's need.

Reputation assessment is also popular in real world applications, for example, in eBay [53] where, the service consumer with the winning bid has an opportunity to rate the transaction with the service provider as positive (+1), neutral (0), and negative (-1). Based on the collected ratings, an aggregated reputation value is calculated as the sum of ratings over the last six months. This simple reputation system has been empirically evaluated in [46, 155, 144] and has been shown to induce trust within the market that helps facilitate service selection. eBay's system however suffers from a known problem of retaliation [155, 182]. Since rating in eBay is bilateral, in that service providers and consumers rate each other, service consumers may be pressurized to rate a service positively as they do not want to receive a negative rating from a service provider as retaliation. eBay allows contextual information to be added to a given rating as a short text description. However, such information is only useful for human consumption and cannot be used for automated processing in an SOC

environment.

Zacharia and Maes [212] presented a reputation system implemented in the Kasbah marketplace [37] which allows service consumers or service providers to build their reputation based on ratings given to each other. The focus of Zacharia and Maes's work is to answer how a reputation system can adapt to change in a service provider's behaviour. For example, their algorithm uses a dampening factor, which associates higher importance to newer ratings and the older ratings are slowly forgotten so diminishing their effect as time passes. This allows services that have improved from a non-favorable performance a chance to compete again in the service selection process. However, this work does not collect or use context behind the reputation ratings and thus is limited in performing effective service selection. Also, this work does not consider QoS assessment.

As part of the ReGreT system Sabater and Sierra [162] presented a social model of reputation [163]. This work attempted to study three dimensions of reputation: individual dimension - service experience based on direct interaction with the service provider, social dimension - information gathered from other members in the society through a social network, and ontological dimension - this considers reputation as a multi-faceted concept that is context dependent. The ontological dimension is represented using a graph structure, for example, the reputation of a service provider as a 'swindler' may be based on 'quality swindler' and to 'overcharge' but not be related to 'late delivery'. This approach makes an attempt to model aspects of the context behind the reputation but lacks explanation on how QoS can be supported or how the context can be used for service selection.

Recent approaches to trust and reputation are based on techniques to aggregate collected ratings about a service provider using well established probability theories. Mui et al. [128, 127] present a Bayesian model [146] for trust and reputation calculation. This system requires ratings given to an entity (e.g., a service provider) to be binary (i.e., positive (1) or negative (0)) and is based on statistical updating of the beta probability density function (PDF). A PDF provides an aggregated view of all ratings and is expressed between 0 and 1, which represents the degree of trust in a service. Here, 0 represent no trust and 1 complete trust that a future interaction with the service will be successful. This model has been very popular and adopted by many other research studies for trust and reputation formalization [84, 197, 200, 183, 153].

Yu and Singh [208, 209] proposed a model of trust and reputation that aggregates service experiences based on the Dempster-Shafer theory of evidence [146]. This theory allows trust to be represented such that lack of belief in a service to be trustworthy does not imply disbelief, which is referred to as the state of uncertainty. An initial uncertainty about a service being trustworthy is replaced with belief or disbelief as trust ratings are accumulated about their performance. The benefits of these probability models lie in their strong mathematical foundation, though such models can be very complex to implement and understand. These studies are based on aggregating trust ratings that do not include the contextual information about the ratings, and thus are limited in performing effective service selection. However, such trust or reputation rating may be used in combination with QoS ratings to take into account, for example, raters trust.

The need for reputation systems that incorporate QoS based contextual informa-

tion alongside the ratings has been identified only recently. Maximilien and Singh [121, 122] presented a conceptual model of reputation for Web services. This work extends services in an SOC environment with a Web Service Agent Proxy (WSAP) that acts as a proxy for clients (service consumers) of the Web services. WSAP performs service selection on behalf of a service requester by consulting external reputation and endorsement agencies to help find appropriate providers. WSAP records feedback on various QoS attributes. Some of these attribute ratings (e.g., price and delay) are derived automatically while others require human intervention. WSAP shares this rating data with external reputation and endorsement agencies to maintain public knowledge about service behaviour. WSAP calculates the reputation of a service as a simple weighted average of all the available ratings. The weights represent the user preference for the weighted QoS attribute(s). Though this model defines use of QoS attributes along with reputation for service selection, it does not include user expectations as part of its context, and thus does not help identify the reasons behind the ratings given by users, which may lead to inaccurate assessment of a service in a user's context [48].

2.2.2 SOC, Workflow and QoS

The problem of Web service selection has also been considered in the area of workflow composition and execution [116, 143, 60]. Workflow research in SOC deals with selecting and composing existing Web services to realize required business processes. The Workflow Management Coalition(WfMC) [41] defines workflow as "the automation of a business process, in whole or part, during which documents, infor-

mation or tasks are passed from one participant to another for action, according to a set of procedural rules. A lot of research [196, 216, 52, 93, 28, 47] has been undertaken into supporting workflow composition. This includes well known projects such as METEOR [33], Crossflow [77], DySCo [149], Self-Serv [19], eFlow [34] and CMI [165], all of which support some aspects of QoS. Most projects focus on QoS analysis, estimation, optimization and monitoring to perform QoS aware service composition.

Projects that have presented initial attempts to create QoS aware service composition, include eFlow [34] and Crossflow [77]. In eFlow, a composite service is described as a process schema that is composed of other basic or composite services. It consists of service nodes that represent the invocation of a basic or composite service; decision nodes that specify the alternatives and rules controlling the execution flow, and event nodes that enable service processes to send and receive several types of event. A service process broker is an eFlow component that is responsible for executing service selection rules and that enables the selection of appropriate services. This selection is based on only one QoS attribute: execution time of the services. This is to ensure that the required total execution time of the workflow is met. The CrossFlow project provides support for workflow management in dynamically established virtual enterprises. This involves contract establishment that will define the business relationship, dynamic infrastructure creation for enactment of the provided service and workflow management between the business partners. The role of QoS in workflow management is to (i) *verify* that the outsourced workflow part performs correctly with regard to the constraints agreed in the contract, (ii) decide on appropriate *reactions* in case deviations occur, and (iii) *predict* the future behaviour of services. The prediction is

based on performance models given as continuous-time Markov chain (CTMC) and produced by an offline monitoring component which analyzes past executions of the workflows or an online monitoring component that maintains the current state of the workflow. In case of significant QoS deviation, the prediction component initiates a replacement service selection in order to maintain the QoS goals. Here, the QoS goals of workflow execution relate to the time and cost involved. eFlow and CrossFlow are workflow management systems that support service selection according to given quality constraints (e.g., time or cost). These approaches however offer limited support for QoS and are restricted to the time attribute, which is only one of the QoS dimensions that may be used [29, 125]. Similar work exists [42, 89] on QoS support for workflow composition, which has concentrated its efforts on single QoS dimension of time.

Much work [202, 193, 113, 191, 216, 31, 107, 73, 129] has been done in providing middleware to support QoS in service composition. Zeng et al. [216] presented AgFlow, a middleware platform that enables the quality-driven composition of Web services. In AgFlow, the QoS of Web services is evaluated by means of an extensible multidimensional QoS model, and the selection of services is performed in such a way that it optimizes the QoS for the composite service selected for a given set of user requirements from the available component services. AgFlow can adapt to changes that occur during the execution of a composite service, by revising the execution plan in order to conform to the user's constraints on QoS. QoS driven service selection is presented using two approaches, one based on local optimization and the other on global planning. The local optimization approach performs optimal service selection

for each individual service in a composite service without considering QoS constraints spanning multiple services and without necessarily leading to optimal overall QoS. In local optimization, a quality vector is computed for each of the candidate services, and based on these quality vectors, the system selects one of the candidate services by applying a Multiple Criteria Decision Making (MCDM) process [214]. The global planning approach on the other hand considers QoS constraints and preferences assigned to a composite service as a whole, and uses integer programming to compute optimal plans for composite service executions. This model provides flexibility by allowing a quality vector to contain any number of QoS attributes based on the Quality ontology and designing this to be independent of the underlying service selection algorithms. This approach also allows user defined weights to be applied to each QoS attribute, thus allowing some form of personalized service assessment and selection. However, this approach aggregates all available monitored data about a service in the assessment. As such, it ignores the fact that some monitored data is based on service expectations which may not be relevant to a particular QoS assessment request and this can result in inappropriate service assessment and selection [48].

2.2.3 Recommender Systems

Recommender systems use the opinions of a community of users to help individuals in that community to identify the content of interest from a potentially overwhelming set of choices. Recommender systems in an e-commerce environment are used to suggest products and to provide consumers with information to help them decide which products to purchase [76]. The most widely used technique for recommenda-

tions both in research and in real-world applications is based on collaborative filtering [154, 75, 123, 85, 7, 18, 126]. Collaborative filtering is defined as "the process in which people collaborate to help one another perform filtering by recording their reactions to documents they read" [61].

Early work on collaborative filtering was presented by Goldberg et al. [61] as part of the Tapestry system. This system proposes a collaborative filtering method for the domain of emails and news. The proposed approach allows user to create queries that allow filtering of items based on assertions (e.g., important, outstanding, excellent) made by other members in the community. A user can thus form a query, for example "show me all emails that Joe found to be important". This system introduced a revolutionary new direction in personalized information filtering. However, users are required to have some explicit knowledge about the members whose assertions (evaluations) are to be used as recommendations. To overcome this restriction an approach known as "Automated Collaborative Filtering" (ACF) was introduced in the GroupLens project [154]. This system helps users find articles they may like in a large stream of available articles. This approach attempts to identify similar users of the newsgroup by comparing their ratings of news articles. The rating algorithm is based on the heuristic that people who agreed in the past will probably agree again, the ratings from users that are found to be similar to a particular user are aggregated to determine whether the articles may be of interest to this user. Similar approaches to recommendations exist in the areas of Movie recommendation (MovieLens[126], [18, 118, 39, 95]), Music recommendation (Ringo[167],[110]) and Web recommendation ([15, 95, 180]).

Most research in recommender systems is based on the GroupLens model which involves the use of an algorithm to first establish similar users and then identify items for recommendation. Collaborative filtering algorithms may be classified as either "memory-based" [154, 167, 17, 172, 91, 210] or "model-based" [145, 156, 102, 39, 207, 111]. The former aim to identify a set of users (i.e., a group) that share similar interests (e.g., users who have purchased the same items from Amazon). A set of items is then recommended to a user based on aggregated ratings of items from these similar users. Memory based algorithms operate over the entire user database to establish similarity. The model based approach analyzes user behaviours to discover a model. A model consists of a number of classes and similar users are grouped together in a class based on their rating behaviour. These pre-computed models are used to make recommendations by simply linking a given user to one of the classes. The most common algorithms for a memory-based approach are either based on correlation [154, 167, 210] or vector similarity [105, 27, 172, 91]. While model-based algorithms are rooted in probability theory. Common probability approaches include Bayesian theory [145, 156, 22, 211], Neural network [102, 39, 130] and cluster models [207, 111, 206, 59]. These types of algorithm have been evaluated in research studies [27, 147, 76] and the results show no clear winner. The evaluation results suggests that an algorithms performance varies with the particular conditions and the best algorithm is not clear in all situations. This has led to hybrid approaches [18, 62, 30, 102, 109] being proposed recently, which try to unify the existing approaches. However, it is very difficult to measure the real benefit of such approaches due to the lack of agreement on assessment criteria [76]. From the service selection point

of view, all these algorithms may help a service consumer receive recommendations of services from users that share similar rating patterns. However they suffer from two disadvantages. First, the item recommended is not based on the current service requirements but represents something that may or may not be part of a future service requirement (e.g., a user requires a movie service, and the recommendation includes an instance of news service). Second, the similarity assessment is performed based on users sharing a similar rating pattern, but similar ratings may not necessarily mean the context within which the ratings are given is the same. Contextual (e.g., QoS) information may help justify the reasons behind a rating and bring confidence to any similarity assessment techniques [75].

2.2.4   QoS and Business studies

Most approaches to QoS incorporated in service selection require a QoS model based on which a service is assessed, and this is a recent area for research [33, 107, 73, 129]. However, answering "what quality is and how it should be assessed" is well researched in the area of business studies [140, 160, 72, 20, 92, 168, 173, 64, 79, 161, 104].

Two directions to modeling QoS in business studies exist. The first looks at quality and any quality assessment having the aim to increase profit for the business [65, 160, 72, 92]. Thus, this approach aims to answer a question such as 'Will there be an increase in profits by increasing the availability of the service'. While the other attempts to understand how QoS may help improve a user's perception of quality [140, 141, 20]. This is an active research in the area of marketing [32, 79, 64, 138]

and consumer behaviour [157, 168, 173] and the former is in the area of operation management [176, 161, 104] and finance [160, 72, 92]. The marketing approach focuses on understanding consumer expectations and applying this in quality assessment. Including consumer expectations as part of the quality assessment is relevant to our study to identify the reasons behind a quality assessment. A finance based approach is purely for the benefit of service providers and does not help in understanding a consumer's view of quality.

One of the most cited works and the basis on which all other works are built in the theory of QoS modeling in business marketing is proposed by Parasuraman et al. [140]. This work is based on a model called "Gap analysis", and defines service quality as the degree of discrepancy between the customer's normative expectations for the service and their perceptions of the actual service performance. The proposed model also describes a comprehensive set of service attributes, that users might use as criteria in assessing service performance. As part of further study, Parasuraman et al. performed empirical evaluation on the Gap analysis model and presented a refined model called SERVQUAL [141]. This model measures service quality along five QoS dimensions: reliability, responsiveness, assurance, empathy, and tangibles. SERVQUAL operationalizes service quality by subtracting a user's expectation scores from their perception scores. The SERVQUAL expectation refers to the service level that customers believe they "should" get from the service provider [137, 139]. The SERVQUAL model is continually being revised, refined and reformed. However, its primary QoS measurement technique has remained unaltered [136, 139, 138, 142]. Even though the research work is slightly dated, SERVQUAL is very popular and

constantly applied by various researchers [189, 81, 14, 11, 198, 199, 175, 99, 170] to numerous service industries as a means of gauging service quality.

SERVQUAL provides a mature and widely accepted model that may be used as a basis for QoS assessment of services for selection purposes. The approach realizes the importance of using context behind the rating, i.e., expectation, for calculating service quality. However, the quality assessment is not personalized for the end user, that is a single quality measure is created for all users consuming the assessment information. This is appropriate within a context of a business organization where all users would represent a shared expectation based on the organization goals. In an SOC environment, however, a service assessment request would be required by many organizations and consumers with different expectations use different service providers. Thus, the assessment should be based on personal expectations.

2.2.5 Summary of QoS approaches for service selection

In this section, we described the related work in service selection. The summary is presented in Table 2.3, which highlights the main features of each of the approaches and analyses their relevance and limitations when applied to service selection.

As can be seen in Table 2.3, most of the approaches provide support for service selection by either using consumer ratings or past historical data. These approaches are great candidates for QoS enhanced service assessment, but they either fail to take account of the reasons behind a consumer's likes/dislikes of a service or ignore the fact that different users can have different expectations from a service and thus the QoS assessment should reflect this diversity. The approach presented in this thesis aims to

	Main Features	Relevance to Service Selection	Weaknesses in supporting Service Selection
Reputation and Trust	Determined based on ratings that represents consumers satisfaction with the service provider. Ratings can represent both subjective and objective attributes	Aggregated ratings represent reputation or trust score for SPs, which can rank them for service selection	Ratings are created and shared without including the reason behind them. Reputation scores are not personalized to service requesters.
Workflow QoS	Autonomous monitoring of conformance that verifies if a service provider adhered to the planned workflow schedule. Optimization of workflow QoS from list of available tasks in terms of time and price	Historical conformance data about the past executions of services involved in the workflow can be used to create QoS rating for the services, which can be used to assist service selection.	Limited QoS attributes are considered and users' views are not considered for QoS assessment, thus limiting applicability to service selection.
Recommender System	Personalized recommendation of services based on consumers' preferences.	Services are ranked based on how well a service fits a user's profile. A service selection can be made from the ranked list of recommended services	QoS is not considered. No historical service provision patterns are evaluated to avoid recommending troubled services.
Business Studies	QoS assessment and ratings represent consumers perspective. QoS assessment models can be applied to a variety of domains	Ratings can be aggregated to rank service providers based on overall consumer satisfaction, which can assist service selection	Limited application to SOC environment. QoS assessment is not personalized based on service requesters' expectation on the service.

Table 2.3: Service Selection Approach

provide QoS enhanced service selection that understands the context behind a user rating, and use this to create personalized QoS assessments for service requesters.

2.3 Summary

In this chapter, we gave a brief survey of existing and ongoing QoS frameworks and research projects most relevant to service discovery and selection. Our analysis of approaches which support QoS discovery suggested that either they do not support a domain independent model for QoS representation or they lack essential features, for example, an ability to represent a consumer's QoS requirements, or they lack applicability to service discovery, in that they lack of matchmaking component that would allow service capabilities to be matched with service requests. This thesis aims to design a framework that overcomes such limitations by providing a domain-independent model which supports service discovery.

The analysis of approaches relevant to QoS enhanced service selection suggested that either they do not support a formal model of QoS as a basis for service selection; or they lack essential features, for example, service assessment that understands the diversity of the consumer's expectations, and thus the need for customized assessment. The focus of this thesis was to design a QoS enhanced service selection technique that not only understands the importance of a consumer's expectations but also caters for such functionality within the SOC environment. In the following chapters we will present our solution to dynamic QoS-augmented service discovery and selection.

Chapter 3

Framework

3.1 Introduction

The research in SOC has resulted in an innovative architecture to represent software components as independent services that can be discovered automatically, and perform a binding operation with other software. Bound services can be executed without knowledge of their underlying platform implementation. This vision provides a great opportunity for businesses where different services may be combined to provide better or new services to the market. Although research into SOC standards has been beneficial, it is far from complete in order to help service requesters in making appropriate decisions based on QoS to discover and select appropriate services.

In this chapter, we describe the design of our QoS incorporated service discovery and selection architecture. This architecture is illustrated using a Virtual Organization (VO) formation scenario [184] in an SOC environment. The content of this chapter is organized as follows. In Section 3.2 we introduce the agent-based VO for-

mation scenario. We also describe a sample use case which will help explain how the components presented in this thesis work. Section 3.3 presents our QoS incorporated service discovery and selection architecture. Section 3.4 describes details of the components involved in the service discovery phase. Section 3.5 describes the components involved in QoS assessment of discovered services for selection. Section 3.6 presents a sequence diagram to help provide a complete picture of interactions between all components in the architecture. Finally, Section 3.7 provides a summary of the chapters findings.

3.2 An Agent based Virtual Organization Formation Scenario

To help better understand the architectural components of the proposed QoS incorporated service discovery and selection system, it is useful to understand the context within which it will be applied. In this section, we describe the role of the developed components in an agent based VO formation scenario [133]. The VO scenario is intended to demonstrate the application of service discovery and selection in an SOC environment.

Agents are software entities that carry out some operations on behalf of a user or another program with some degree of independence or autonomy, and in so doing, employ some knowledge or representation of the user's goals or desires. Virtual organizations are composed of a number of autonomous entities (representing different individuals, departments and organizations), each of which has a range of problem

solving capabilities and resources at their disposal. These entities co-exist and sometimes compete with one another in a ubiquitous virtual marketplace. Each entity attempts to attract the attention of potential customers by describing the cost and qualities of its services, with the goal of selling these services.

The VO scenario is designed with the aim to *"provide models and techniques for VO management that operate in a robust and resilient manner in complex electronic commerce scenarios"* [184]. More specifically, the VO scenario has the following:

- Autonomous agents to represent the different problem solving entities. These agents are capable of exhibiting flexible problem solving behavior in pursuit of their objectives in uncertain and changing environments.

- Sophisticated interaction models that enable autonomous agents to decide when to try and form a VO, how to actually go about forming the VO, and then to subsequently act as a VO.

- Rich knowledge representation and information inter-change mechanisms that enable agents to describe their aims and objectives to their potential business partners during the formation and ongoing operation of a VO.

- Techniques for modelling QoS that support assessment of agents (i.e., as a service provider) and provides a decision support model for agents (i.e., service requesters) in a VO.

A key objective in putting a VO together is to ensure that they are both agile (i.e., able to adapt to changing circumstances); and resilient (i.e., able to achieve their objectives in a dynamic and uncertain environment). In such environments, the

participants behavior will be informed by exploiting a number of diverse forms of information-advertisements (capabilities and quality of service of individual agents), meta-data (schemas) and information resources (databases and knowledge bases).

To help improve the presentation of an overall picture of the VO formation process, we will use a specific scenario, where a user has certain multimedia requirements for purchasing a service. The scenario is as follows:

Multimedia Scenario: *"A user wants to purchase and receive a monthly movie subscription package on his PDA/phone,and a monthly news service. The user also wants a monthly package for his PDA/phone that includes 30 free text messages and at least 50 free minutes per month".*

This scenario shows the following important characteristics that must be taken into account in the development of an effective VO system:

1. There may be multiple services available from a number of agents representing independent organizations. The diverse agents may offer broadly similar services. The services themselves are described by multiple attributes, for example, price and quality.

2. The services available may change over time, in that new services may become available, or agents may alter the way in which existing services are offered. Services may differ in terms of the number and heterogeneity of the tasks involved in (i) the delivery of the service and their degree of interdependence, and

(ii) the type and frequency of interactions between different customers while the service is being delivered.

3. The agents involved in the system may also employ different policies for dealing with the uncertainty inherent in such a domain; for example, an agent may generate slack resources to limit the possibility of a loss in service to the customer, or it may employ rigorous coordination mechanisms to improve supply chain integration.

The formation of a VO is grounded on three key technologies: (1) the decision-making mechanism of an individual agent, (2) an auction mechanism for the allocation of contracts, and (3) the representation of services. In order to illustrate the operation of VO formation based on these three mechanisms, we present the model shown in Figure 3.1. The model is characterized by the following agents providing various functionality and interacting with each other:

1. Service Provider (SP) Agent: This agent provides services;

2. Service Requester (SR) Agent: This agent represents the consumer of services;

3. Yellow Pages (YP) Agent: This agent provides a registry and a lookup service for services with required capability;

4. QoS Agent: This agent assesses QoS rating for services to determine how well a service will meet SR's expectations; and

5. Clearing Agent: This agent offers an auction mechanism that the SR agent can use to choose an optimal combination of services.
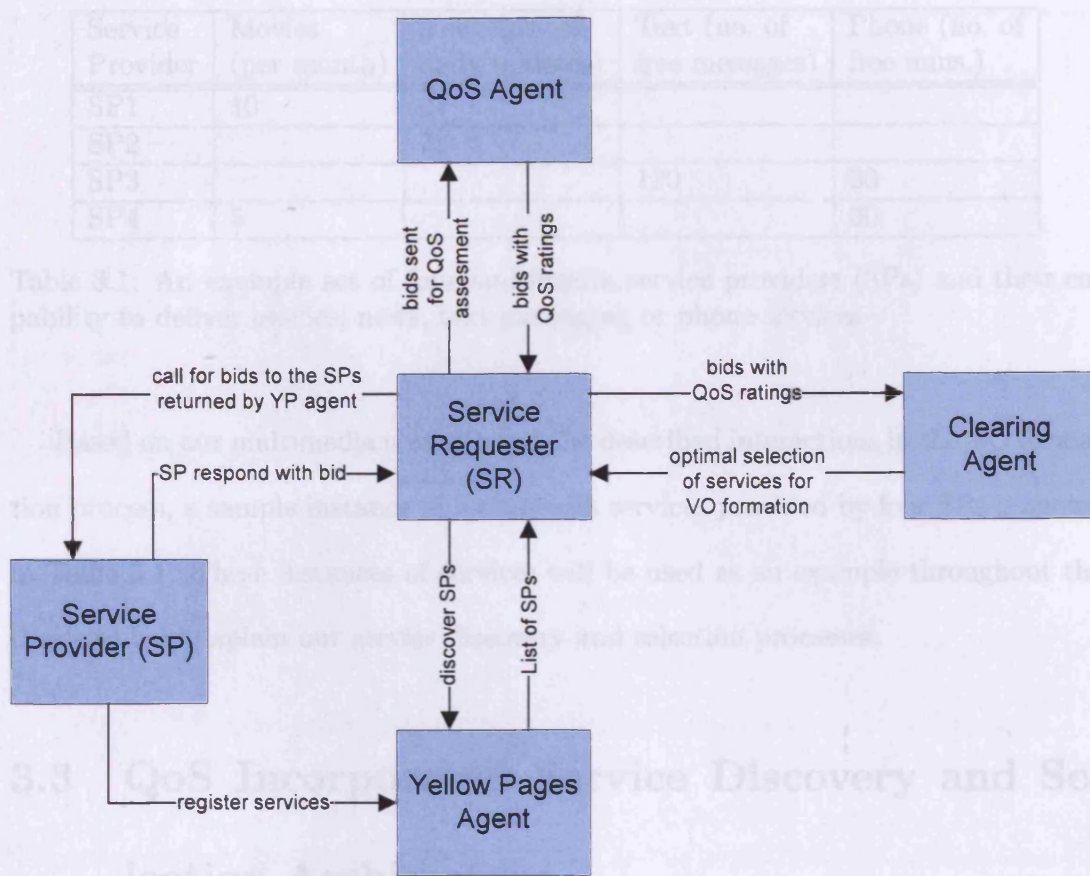
Figure 3.1: VO Architecture

The interactions between agents that results in VO formation can be divided into four main stages - service discovery, bidding, service selection and clearing. Service discovery allows a SR to look up services that can meet certain service requirements. The SR Agent can invite SPs of the discovered services to bid, i.e., offer customized services. The SR agent can then request the QoS Agent to establish QoS ratings for service selection support. Finally, the SR chooses an optimal combination of services to form a VO. In this thesis we consider the discovery and selection steps only.

Service Provider	Movies (per month)	News (no. of daily updates)	Text (no. of free messages)	Phone (no. of free mins.)
SP1	10	24		
SP2		72		
SP3			120	30
SP4	5			30

Table 3.1: An example set of four multimedia service providers (SPs) and their capability to deliver movies, news, text messaging or phone services

Based on our multimedia scenario and the described interactions in the VO formation process, a sample instance of multimedia services provided by four SPs is shown in Table 3.1. These instances of services will be used as an example throughout the thesis to help explain our service discovery and selection processes.

3.3 QoS Incorporated Service Discovery and Selection Architecture

In this section we describe the design of QoS incorporated service discovery and selection that is influenced but not limited to the VO formation scenario (Section 3.2). There are different agents in the VO formation scenario, but the main entities that are relevant to our study are the YP agent and the QoS agent. Figure 3.2 shows a detailed view of our QoS incorporated service discovery and selection architecture.

Conceptually, the YP and QoS agents are exposed as independent, *"trusted"* entities that are neither influenced nor controlled by any other agents in the environment. The element of trust is very important as this states that these agents would not hide or modify information for either intentional harm or for its own personal benefit.
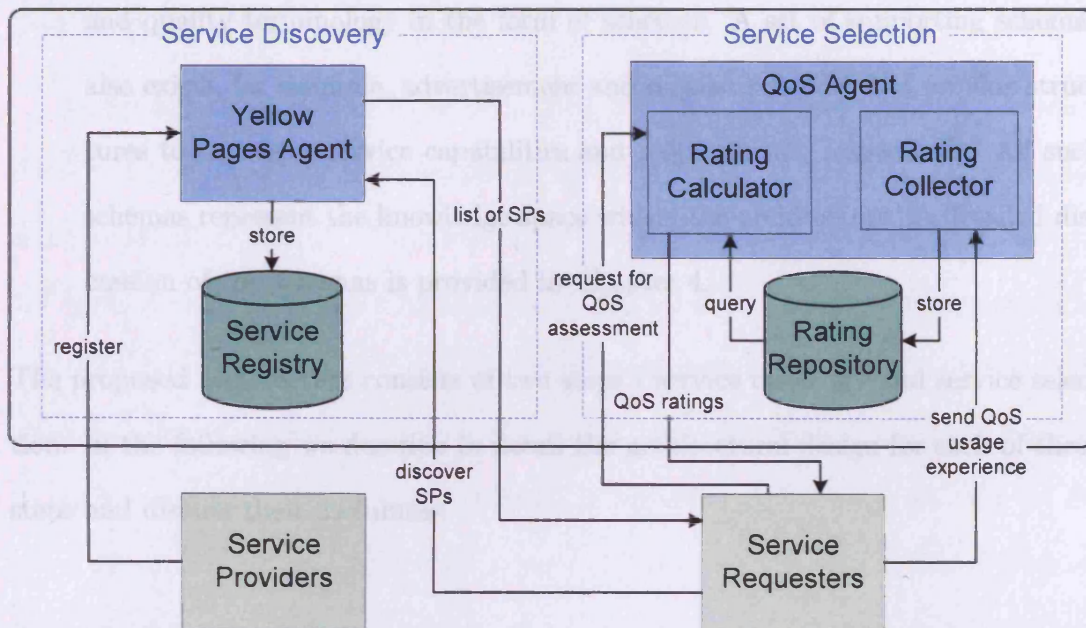
Figure 3.2: QoS Incorporated Service Discovery and Selection Architecture

These agents are also designed to be *"public"*, in the sense that any agent within the environment can communicate with these agents. In the following we describe the two components of our proposed architecture that supports the functions of YP and QoS agents.

- Agent Interface: Each agent includes an interface to allow communication with external entities. This component provides a messenger service that either listens to new messages and performs appropriate actions or sends a query/response message when needed.

- Knowledge Space: All agents present in the QoS incorporated service discovery and selection architecture share a common conceptual understanding of service

and quality terminology in the form of schemas. A set of supporting schemas also exists, for example, advertisement and request schemas that provide structures to represent service capabilities and requirements, respectively. All such schemas represent the knowledge space within the architecture. A detailed discussion of the schemas is provided in Chapter 4.

The proposed architecture consists of two steps - service discovery and service selection. In the following we describe in detail the architectural design for each of these steps and discuss their usefulness.

3.4 Service Discovery

The components involved in the service discovery phase are discussed in this section. Interactions between the agents and assumptions made to achieve the service discovery task are also presented.

3.4.1 YP Agent

This component performs the "matchmaker" function to connect a SR agent with an SP agent who can fulfill certain capability requirements. To perform such a matchmaking task, the YP agent communicates with SR and SP agents. The YP agent interacts with SPs for service advertisement related tasks and with SRs to provide the service discovery facility. Such interactions can be divided into two main stages - registration and matchmaking.

Registration Stage

An SP agent initiates an interaction with the YP agent to advertise one or more services it can offer. The message content for this interaction is based on a predefined schema provided as part of the knowledge space. In this case the schema used is an "advertisement" schema. In addition to this schema, an SP agent also needs to use the available service and quality schemas for forming the advertisement. The schemas allow a standard set of terminology to be established for the YP agent to understand and process the advertised information. It is assumed that an SP agent is capable of creating service advertisements that conform to the predefined schemas. On receipt of any service advertisements, the YP agent parses them to check their validity. An appropriate advertisement acknowledgment is returned to the SP agent after the validity check. It is assumed that the SP agent is truthful in its advertisement, i.e., the information supplied reflects the actual capability that the SP has got to offer. It is possible to involve an external certification authority who can verify and certify an SP's advertisement, but our aim is not to validate the service that an SP claims to offer, but to assess its capability to meet QoS claims as part of the service selection process. An SP agent can send one or more advertisements to the YP agent and can advertise services in any domain. It can also include both functional and QoS descriptions in its advertisements. For the rest of the thesis we refer to attributes that define certain core service capabilities as functional attributes. QoS attributes, on the other hand, represent non-functional attributes of a service. The list of functional and QoS attributes are domain dependent and an exact list of attributes will be defined in a service and the chosen QoS ontologies used in the service

discovery process. The functional description of services in the multimedia scenario would include, for example, movie service of type "horror" and QoS attributes will include attributes such as "frame rate" for a movie. How advertisements are created will depend on the availability and use of supporting tools. Figure 3.3 displays three possible interaction design for SP agents to create service advertisement: (a) Manual: a user manually creates the service advertisements by entering all the required details using the available schemas; (b) Semi-automatic: an advertisement template tool is available that provides a form for a user to fill in. On submission of the form, the required advertisement structure with user input values is created automatically; and (c) Autonomous: an advertisement template tool interacts with business components of the SP to generate an advertisement automatically. For example, a resource and business modelling components may observe the current status of available resources and access pricing policy to generate advertisements [131, 45]. In our model, a manual approach to creating service advertisement is used as our primarily focus is on the semantics of the advertisement and how matchmaking is performed based on the semantics.

Matchmaking Stage

An SR agent can request the YP agent for services that can meet certain capability requirements. The request is based on the "request schema", which is part of the knowledge space. The steps to define the content of a service request message by an SR are similar to those used for advertisement. The only difference is the heading of the message; "request" instead of "advertisement". This is used to help identify
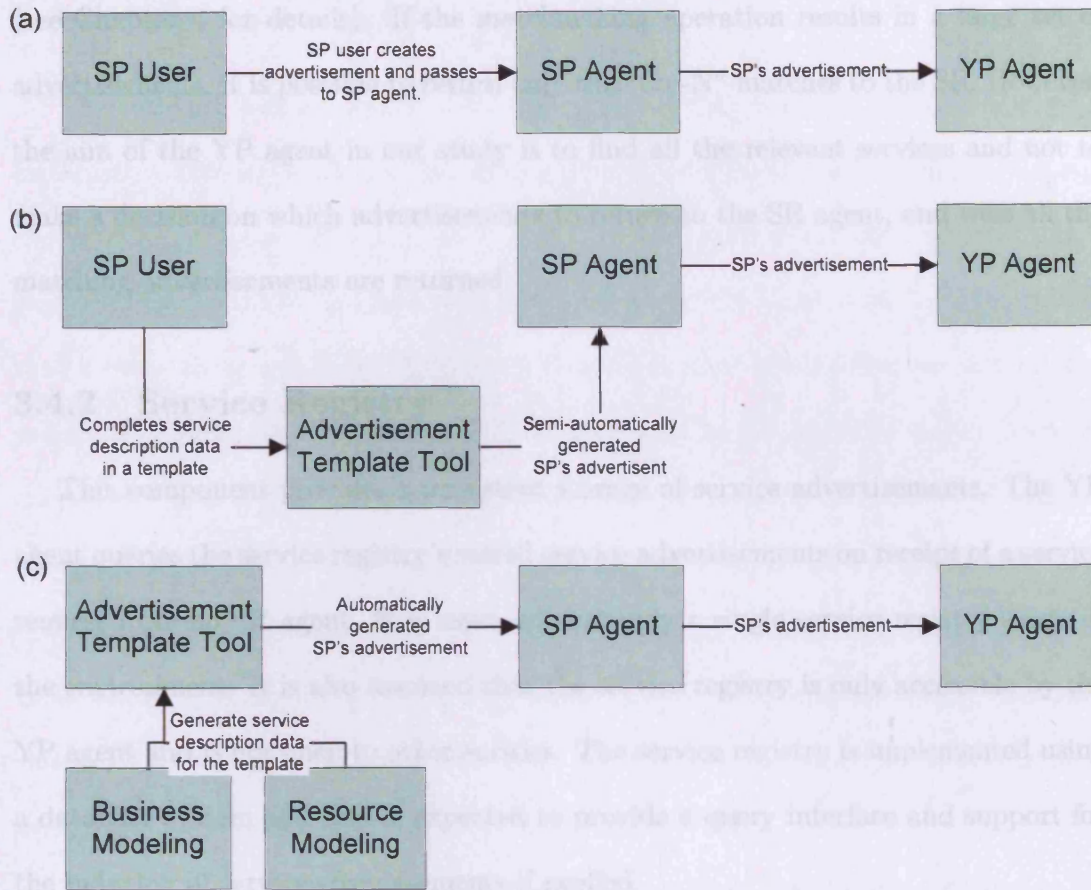
Figure 3.3: Creation of advertisement

the message for appropriate processing by the YP agent. This allows matchmaking between a request and an advertisement based on mutually understood terms. Similar to the advertisement, we expect a user to manually create a service request by entering all the required details using the available schemas.

The role of the YP agent on receipt of a service request from an SR is to respond to it with a list of SPs which can provide the functional and QoS requirements specified in the request. The YP agent uses a matchmaking algorithm to perform this operation

(see Chapter 4 for details). If the matchmaking operation results in a large set of advertisements, it is possible to return only the "top-N" matches to the SR. However, the aim of the YP agent in our study is to find all the relevant services and not to make a decision on which advertisements to return to the SR agent, and thus all the matching advertisements are returned.

3.4.2 Service Registry

This component provides a persistent storage of service advertisements. The YP agent queries the service registry's stored service advertisements on receipt of a service request from an SR agent. It is assumed that only a single service registry exists in the environment. It is also assumed that the service registry is only accessible by the YP agent and is not open to other entities. The service registry is implemented using a database system and thus is expected to provide a query interface and support for the indexing of service advertisements if needed.

3.5 Service Selection

Central to the service selection process is the QoS agent. The QoS agent is composed of two sub components - a rating collector and a rating calculator. In addition to the QoS agent, a repository component also exists in the architecture to provide a data store for QoS rating information. We describe these in detail in the following sections.

3.5.1 Rating Collector

The rating collector interacts with any SR agent that is willing to share its service experience. This interaction is based on a predefined schema made available through the knowledge space. The schema in this case has three properties associated with each QoS attribute for a given service, which allow the QoS usage information for that service to be specified. The three properties are: *expectation*, *perception*, and *rating*. The *expectation* represents a QoS level that the SR *hoped to receive from the service provision*. The *perception* specifies *what* the SR *perceived was received when using the service*. The *rating* indicate the *degree of satisfaction the SR has with this provision*. We assume that such QoS usage information is collected from an SR agent. However, such information can also be obtained from SP agents or through proactive monitoring of the service provision. Since our aim is to use this QoS information to inform SRs, we collect data from SRs only.

Figure 3.4 shows the three main types of interaction that may exist between an SR agent and the Rating Collector when gathering QoS information. These are:

- An SR agent gathers QoS usage information completely autonomously without any communication with the user (Figure 3.4(a)). This process provides an efficient rating collection mechanism. It is also free from human errors resulting from mistaken judgements. However, this approach is limited to processing "objective" QoS attributes such as "response time", which may be monitored by tools. Subjective attributes, such as "level of detail" of news, cannot be monitored using tools and thus can only be rated by user feedback.

- The user communicates all QoS usage information directly to the rating collector

based service selection environment. It is assumed that all SR agents share their QoS experiences for mutual benefit. This assumption fits in line with many trust and reputation systems (Chapter 2). Research has been undertaken into providing
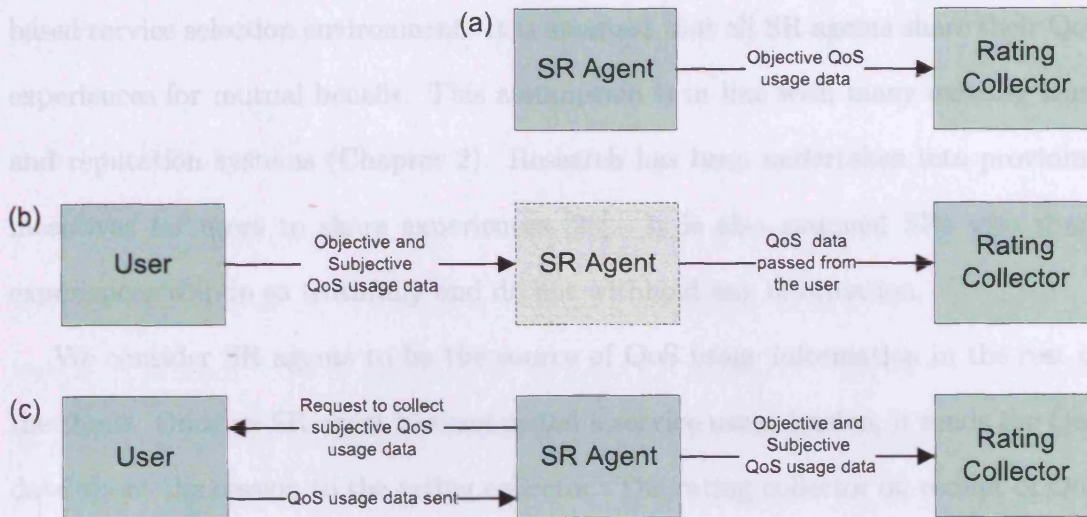
(Figure 3.4(b)). In this case, the role of the SR agent is equivalent to that of a "dummy agent", where a dummy agent can be thought of as a special type of agent that does not contain any intelligence and is only used as an interface to communicate messages. This type allows any QoS usage information to be observed by the user and communicated, but this can be very time consuming for the user and susceptible to human errors.

- In Figure 3.4(c), the SR agent monitors possible objective QoS attributes and requests the user to fill in the remaining subjective information. This process allows a rich set of data on QoS usage to be collected and provides some tradeoff between the amount of user effort required and the degree of error that may occur.

It is worth noting that both the SR agent and the user are outside our QoS

based service selection environment. It is assumed that all SR agents share their QoS experiences for mutual benefit. This assumption is in line with many existing trust and reputation systems (Chapter 2). Research has been undertaken into providing incentives for users to share experiences [26]. It is also assumed SRs who share experiences will do so truthfully and do not withhold any information.

We consider SR agents to be the source of QoS usage information in the rest of the thesis. Once an SR agent has completed a service usage session, it sends the QoS data about the session to the rating collector. The rating collector on receipt of QoS data, parses it to check its validity (based on a predefined schema). If valid, the newly acquired information is sent to be stored in the rating repository. It is possible that a more complex form of interaction style could exist between the SR agent and the rating collector. For example, when using a data stream based approach, where the SR agent would send any QoS usage information during service provision, as soon as they become available. This would allow the most up-to-date information to be made available for QoS assessment, but it requires the maintenance and processing of a large amount of data which may also be complex[178].

3.5.2   Rating Repository

The rating repository acts as a persistent store for the SRs' service usage experience. The rating collector stores the received QoS usage information in the rating repository. The rating calculator queries this repository to access QoS usage information for a particular service. The service repository like the rating repository is implemented using a database system that provides a query interface. The rating

repository is a centralized data store for all the QoS usage information.

3.5.3 Rating Calculator

The rating calculator provides a QoS assessment service to the SR agents. It processes the QoS usage information present in the rating repository to discover the history of a service and to establish an overall QoS rating for that service. The QoS rating calculation is initialized on receipt of a request for QoS assessment from an SR agent. Such a request may be thought of as a bid which is similar to a service advertisement, describing a service offered by an SP that includes functional and QoS information. The rating calculator establishes a QoS rating for each service bid received from an SR. It is assumed that each bid is annotated with the SR's QoS expectations for the service. An expectation is a value associated with a QoS attribute of a service in the bid. For each QoS attribute, the SR's expectation value can be either the same as the QoS attribute and value pair in the bid, or higher/lower than the bid value, or a value may not be specified in the bid. For the first two cases, the expectation value can be considered as being influenced either by the claims made by the SP in its service advertisement; the SR's past experience(s) with the service, or by some recommendation from a third party. Given that our objective is to use the SRs provided expectation to create a rating for a service, the method used by an SR to derive its expectations is outside the scope of this thesis, as it will not affect how it is used to calculate the ratings. When an expectation value is included in the request it is utilized in the QoS assessment for service selection. This usage will not be affected by its method of derivation.

Based on the expectation of a service given by the SR agent, a subset of the QoS usage information for that service is retrieved from the rating repository. The rating calculator performs its QoS assessment by processing the retrieved information to generate a QoS rating for the service. If no data is returned by the rating repository, a default QoS value is assumed (e.g., null).

3.6 Interaction between the components in the architecture

The UML sequence diagram Figure 3.5 illustrates the interactions between the various components in the service discovery and selection phases. The primary component in the discovery phase is the YP agent and in the selection phase it is the QoS agent. The interactions represent a transfer of messages between the components. All the agents are bootstrapped together using an agent platform. Once bootstrapped, the agents are aware of each other's existence and can communicate with each other. The interactions that occur in the two phases are:

- The SP agent registers its services by sending service advertisements to the YP agent. Assuming that the advertisements conform to the advertisement schema, the YP agent returns an acknowledgment that contains "advertisement ids" to the SP. The YP agent stores the service advertisements in the service registry.

- The SR agent then sends a request to the YP agent to find services that can meet certain capability requirements. On receipt of a service request, the YP agent executes the matchmaking algorithm that matches the *request* with the
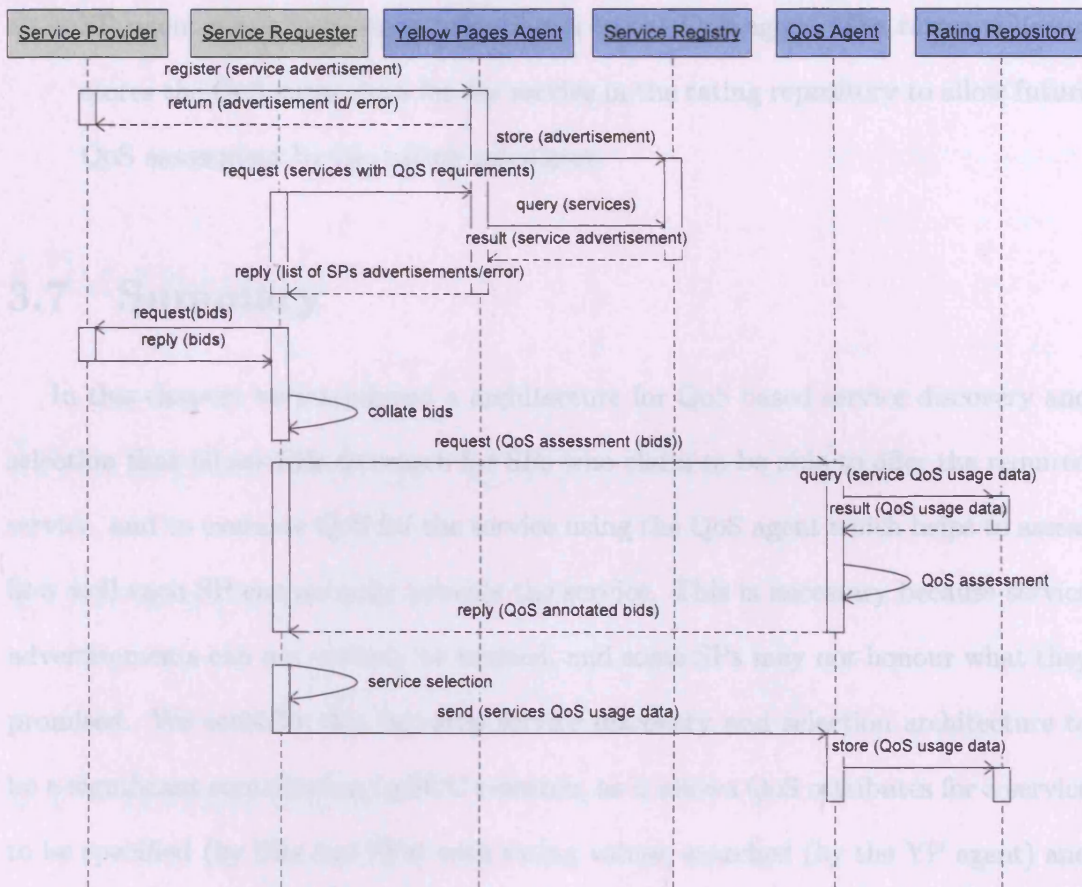
Figure 3.5: Sequence diagram representing the service discovery and selection processes

adverts by querying the service registry. The YP agent then sends a list of SPs that satisfy the service request to the SR agent.

- After the list of SPs advertisements is received by the SR, the SR asks for bids from those SPs. Once the bids are received from the SPs, the SR sends the bids to the QoS agent for QoS assessment. The rating calculator of the QoS agent will assess each bid and return a QoS rating for the service to the SR agent. Finally, the SR agent decides on which SPs to use. After using the service, the

SR agent sends QoS usage information to the QoS agent. The rating collector stores the QoS usage data for the service in the rating repository to allow future QoS assessment by the rating calculator.

3.7 Summary

In this chapter we introduced a architecture for QoS based service discovery and selection that allows SRs to search for SPs who claim to be able to offer the required service, and to evaluate QoS for the service using the QoS agent which helps to assess how well each SP can actually provide the service. This is necessary because service advertisements can not entirely be trusted, and some SPs may not honour what they promised. We consider this two-step service discovery and selection architecture to be a significant contribution to SOC research, as it allows QoS attributes for a service to be specified (by SRs and SPs) with rating values, searched (by the YP agent) and evaluated (by the QoS agent). This approach provides a more meaningful architecture to support service discovery and selection in SOC.

Chapter 4

Incorporating QoS in Service

Discovery

4.1   Introduction

In this chapter we describe the semantics needed for defining QoS. Also, we present how QoS is incorporated into dynamic service discovery within a SOC. Service discovery in an SOC environment is an important issue because the environment can potentially contain thousands of autonomous services which may enter or leave the environment at any time. It is desirable therefore that a user is able to request a service by stating declaratively what is required, rather than by having to specify how to obtain or access a specific one at a pre-determined location. In other words, discovering which services are available to meet the requirement of a specific service request should be performed dynamically at the time of the request.

Current approaches to service discovery in an SOC environment allow match-

making between requested and advertised services based largely on their functional attributes. To illustrate this, consider the multimedia scenario introduced in Section 3.2. Suppose that we have an SR who wishes to purchase a monthly subscription package for science fiction movies and news services, including 30 free text messages and 50 free talk minutes. This service request may be represented abstractly, as shown in Figure 4.1, where the request is made in terms of the required functions.

```
– SubscriptionType = monthly
    – VideoContent:
        mediaStyle = scienceFiction
    – HTMLContent:
        mediaStyle = news
    – TextMessages:
        messages = 30
    – PhoneCalls:
        minutes = 50
```

Figure 4.1: A Request for Service

For example, the qualifying SP must offer `scienceFiction` movies. To determine, which SP(s) can offer the required service, the matchmaking component searches through the registry, typically using a string comparison method, to see if any registered services match some or all of the functional attributes listed in the request. If, SP1 has advertised that it offers `scienceFiction` movies, then SP1 is identified as a potential provider for the required service and its details are returned to the SR.[1]

The above model works well, if we assume that the SR is only interested in the functional aspects of a service. In practice, however, it is quite possible that we may

---

[1]Note that it is not necessary to find a single SP who can satisfy the SR's request completely, and the matchmaking component will search for all the SPs who can serve any part of the request.

have several SPs offering the same service, and just like in a common marketplace, an

SR may wish to select a service based on functional as well as quality. Thus, it would

be desirable that the SR can pose the request shown in Figure 4.2, where `frameRate`

= 24fps and `availability` = 7DaysAweek are the QoS requirements.

Figure 4.2: Service Request with QoS Requirements

The current function-based approaches are not sufficient to support this more advanced form of service discovery. In this chapter, we describe our approach to service discovery that incorporates QoS specifications and requirements into the matchmaking process. The contents of this chapter are organized as follows: Section 4.2 provides the necessary schemas for the knowledge representation of the services and QoS information. In particular, this section includes a description of service and quality schemas. In addition, we also describe how the knowledge representation is used to allow: SPs to advertise their functional capabilities and QoS promises to the service registry, and SRs to request a service by specifying not only functional requirements,

but also QoS requirements. Section 4.3 describes how matchmaking between advertised and requested services based on functional as well as QoS requirements is supported in our approach. Finally, Section 4.4 summarises this chapter.

4.2 Knowledge Space

In Section 3.3, we discussed how different agents communicate with each other to perform service discovery and selection. The content of these communications depends on how services with QoS attributes are described. A standard representation of knowledge allows a consistent set of terminology to be used between all agents in the environment. To enable this, it is necessary to have an expressive service description language and appropriate schemas. We use DAML-S [9] for service description. DAML-S was chosen because it is capable of describing not only low-level service functionality and requirements in terms of its message format, data types, and protocols, but also semantic information such as service classification and descriptions. DAML-S however does not provide support for expressing QoS as part of a service description, thus this will require additional concepts to be defined.

To standardize terminology and distinguish between service and quality terms in service description, we developed two schemas, the *service schema* and the *quality schema*. The *service schema* provides SPs and SRs with a common terminology for advertising and requesting services, and enables the YP agent to match advertisements with requests. The *quality schema*, on the other hand, specifies what the QoS attributes are and how they are related to the services.

4.2.1 Service Schema

A service schema, is created using DAML-S and is relatively straightforward, but is domain dependent. For example, a sample service schema for a simple `media` application is shown in the lower half of Figure 4.3. For simplicity of presentation, we have expressed the schema here as a class diagram, rather than its implementation in DAML-S. The actual implementation of the media application schema used in this work is provided in Appendix A. It is important to realize different service schemas are needed for different domains.
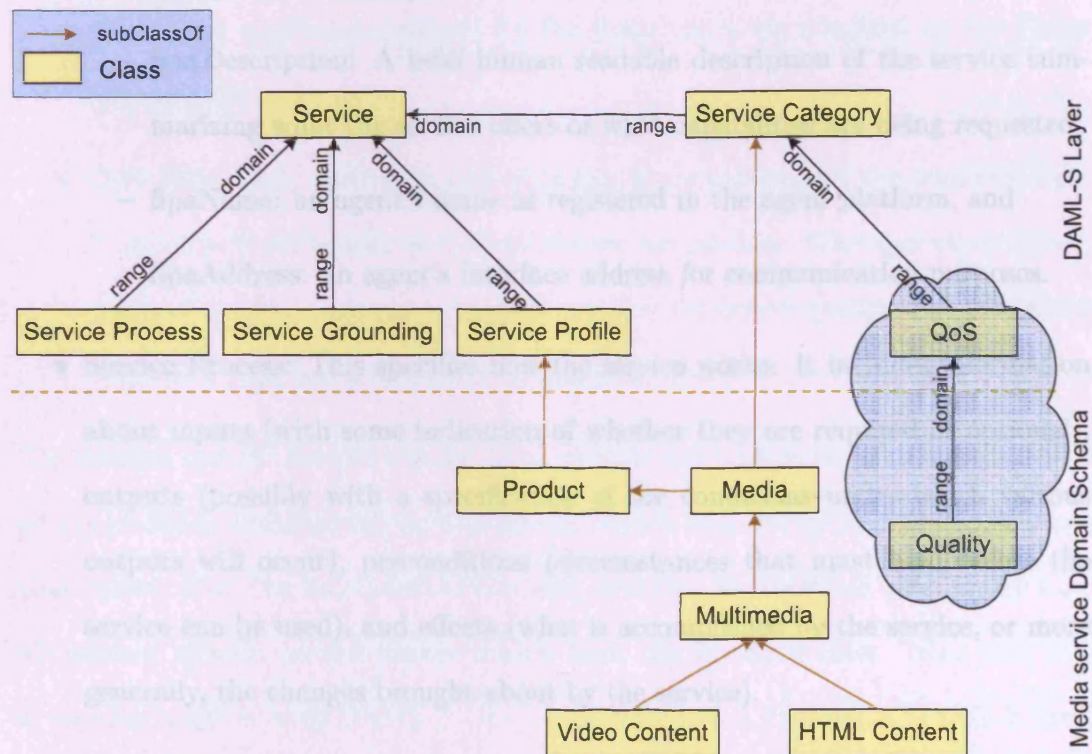
Figure 4.3: Service schema as described by DAML-S and Media Service Domain Schema

In Figure 4.3, a `Service` according to its DAML-S definition consists of three important classes: the service profile, the service process, and the service grounding. Each of these classes plays a different role in service description, namely:

- Service Profile: The profile describes what the service does. It characterizes the service for purposes of advertising, discovery, and matchmaking. That is, it gives the kinds of information needed by the YP agent to determine whether the service meets an SR's needs. More specifically, a profile describes four properties;

    - serviceName: the name of the service,

    - textDescription: A brief human readable description of the service summarizing what the service offers or what capabilities are being requested,

    - fipaName: an agent's name as registered in the agent platform, and

    - fipaAddress: an agent's interface address for communication purposes.

- Service Process: This specifies how the service works. It includes information about inputs (with some indication of whether they are required or optional), outputs (possibly with a specification of the conditions under which various outputs will occur), preconditions (circumstances that must hold before the service can be used), and effects (what is accomplished by the service, or more generally, the changes brought about by the service).

- Service Grounding: This specifies how the service is to be used. Typically a grounding may specify a communication protocol (e.g., SOAP [68]), and service-specific details such as port numbers used in invoking the service.

For our work, we use **Service Profile** only, as this is sufficient for service discovery. In addition to the three classes provided by DAML-S, we define two additional classes: **Service Category** and **QoS**. These classes together allow us to specify additional information about the services. The detailed descriptions of each of these two classes is:

- Service Category: In an SOC environment, services from any domain (multimedia, travel, telecommunication) can potentially be offered by an SP. The purpose of introducing this class is to allow services to be grouped, so that a wide variety of class-specific attributes may be specified for a group. For example, all the attributes specified for the **Media** class are inherited by the **Video Content** class.

- QoS: Figure 4.4, depicts the **QoS** class that is associated with the **Quality** class. It specifies a collection of QoS attributes for services. The **QoS** class defines different quality concepts (objective quality, subjective quality, etc.,) that are relevant to the characterization of QoS attributes.

The domain specific service classes, (e.g., media) are integrated with the DAML-S service profile to facilitate service description. This is achieved by making our **Product** class inherit from the **ServiceProfile** class provided by DAML-S (see Figure 4.3). All domain specific service classes inherit from this **Product** class. Note that due to naming conflicts with DAML-S, our top-level service is **Product** – DAML-S itself defines **Service** as a class from which all services inherit. In the next section, we describe the **Quality** schema that allows QoS information to be incorporated within a service description.

4.2.2 Quality Schema

Representation of a quality schema is currently not supported by DAML-S, and our proposed approach to enabling this is explained in this section. The quality schema (see Figure 4.4) provides the necessary concepts an SP requires to create its service advertisement with QoS and an SR needs to specify its service requirements with QoS. The quality schema consists of two classes: the QoS class and the Quality class. The Quality class inherits from the QoS class. The Quality schema defines three following important concepts that help characterize its QoS attributes:

1. Service Specific and Service Independent: For different types of service, a large number of attributes may be used to describe their QoS attributes. Some are domain dependent and will only be relevant to a specific type of service. For example, frame Rate is only relevant to a movie service. Others are domain independent and are applicable to all types of service, for example, availability [119]. Thus, it is important that our quality schema distinguishes between these two types of QoS attribute, so that we do not repeat ourselves when specifying service independent QoS attributes for each individual service. Motivated by this observation, we group all QoS attributes into *service specific* and *service independent* classes. As can be seen in Figure 4.4, a service must include its service specific QoS attributes explicitly, e.g., VideoContent has frameRate as one of its QoS attributes, but includes service independent QoS attributes implicitly, e.g., VideoContent also has Availability, Performance and Reliability among its QoS attributes.

2. Objective and Subjective: This allows a given QoS attribute's type to be de-

scribed as either *objective* or *subjective*. Some attributes, for example `frameRate` and `Availability`, are considered to be objective as they can be measured accurately by automatic monitoring tools. However, subjective attributes, for example `Completeness` of coverage of news, cannot be measured using any monitoring tool and can only be judged by human users. In Chapter 5 we describe how we use the QoS agent to measure QoS either through monitored data or ratings from the users.

3. Positive and Negative: This allows a given QoS attribute's type to be described as either *positive* or *negative*. A *positive* quality type specifies that a high value is preferrable to a low value. For example, high performance is usually considered to be better than low performance. While a *negative* type specifies that a low value is preferred to a high value. For example, a low response time is more desirable than a high response time.

Based on these concepts, the `Quality` class provides a set of service independent QoS attributes (as shown in Figure 4.4). A sample set of service specific QoS attributes for a simple `media` domain is also shown in Figure 4.4. However, it has to be noted that this list is not intended to provide a complete set of relevant QoS attributes for an SOC environment. It is an example set to demonstrate how QoS attributes can be incorporated into the Quality schema. Many other QoS attributes can form part of the `Quality` class, and in the real world these would depend on the requirements of a domain expert. The set of service independent QoS attributes shown in Figure 4.4 are:

1. **Availability**: represents the degree that a service can be utilised at a specified
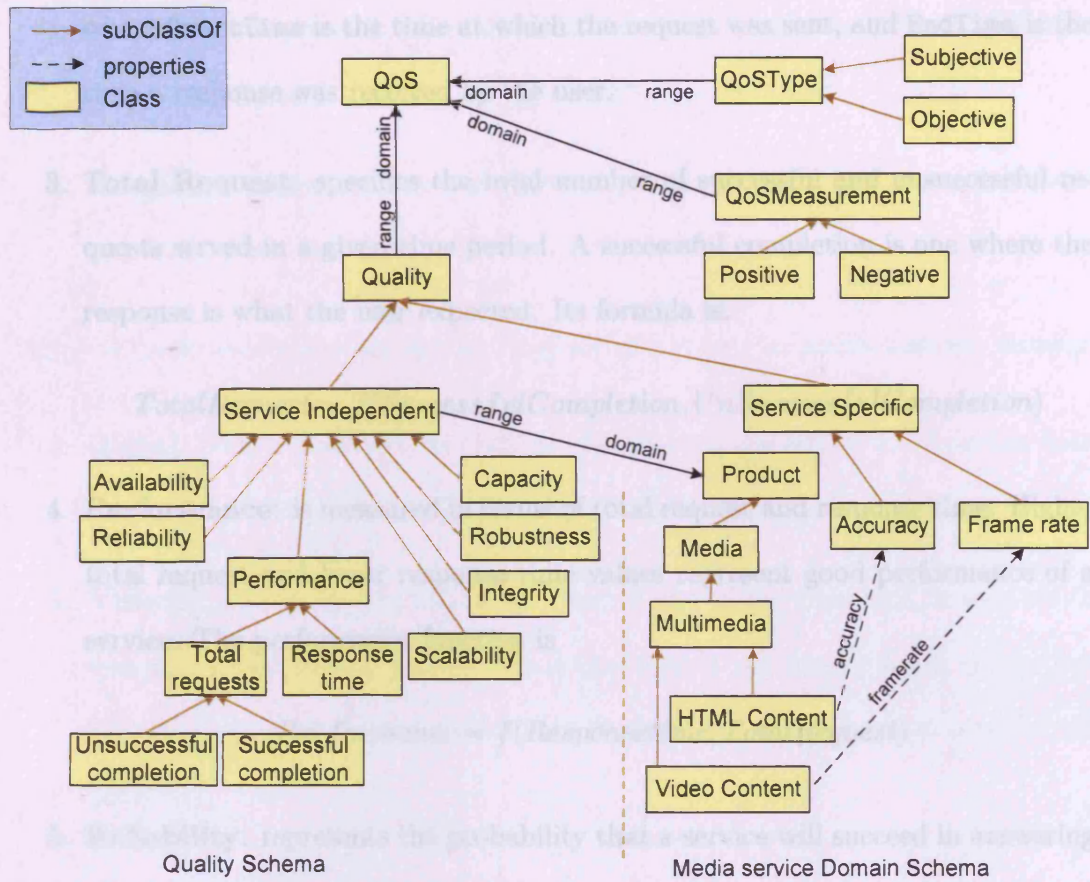
Figure 4.4: Quality Schema

time. The availability category consist of two attributes, `TimeAvailable` and `TotalTime`. The function for computing the value for availability is:

$$Availability = f(TimeAvailable, TotalTime)$$

2. **Response time**: is the round-trip time between sending a request and receiving the response. It is computed as:

$$Responsetime = f(StartTime, Endtime)$$

where `StartTime` is the time at which the request was sent, and `EndTime` is the time a response was received by the user.

3. **Total Request**: specifies the total number of successful and unsuccessful requests served in a given time period. A successful completion is one where the response is what the user expected. Its formula is:

$$TotalRequest = f(SuccessfulCompletion, UnSuccessfulCompletion)$$

4. **Performance**: is measured in terms of total request and response time. Higher total request and lower response time values represent good performance of a service. The performance function is

$$Performance = f(Responsetime, TotalRequest)$$

5. **Reliability**: represents the probability that a service will succeed in answering a request, it promised within a particular response time. Reliability is measured by the number of successful completions for a service within the response time. It is calculated as:

$$Reliability = f(TotalRequest, SuccessfulCompletion)$$

6. **Robustness**: represents the degree to which a service can carry out the functions properly under abnormal circumstances, such as when it is given invalid inputs [201].

7. **Capacity**: is the size/quantity of resources a service can provide in a given time period [66].

8. **Scalability**: represents a service's ability to either handle the increasing amounts of work in a graceful manner, or be readily adjusted [201].

9. **Integrity**: is measured by whether the transferred data between service and user is modified in transit [179].

Figure 4.4 also shows domain specific QoS attributes for the media domain, namely:

1. **HTML News Content Accuracy**: Accuracy is the degree to which the data provided correctly reflects the real world objects or events being described [54].

2. **Video Content Frame Rate**: A frame is one of the many still images which compose the complete video. The frame rate is the measurement of the frequency (rate) at which a video content produces unique consecutive frames [201].

4.2.3 Creating Service Advertisements and Service Requests with QoS

In this section we describe how service advertisements and requests are created based on our proposed *Quality* and *Service* schemas. The concepts relevant to advertisements and requests are described with an example based on the multimedia scenario presented in Section 3.2. SPs and SRs create advertisements and requests by performing two common steps. Firstly, the service schema is used to describe the service name, service category and functional properties of the services. Secondly, the quality schema is used to describe different QoS attributes. It is worth noting that

SPs and SRs do not have to specify values for all QoS attributes that are listed in the quality schema. If a QoS attribute is unspecified, it is treated as *unknown*.

Service Advertisement

Figure 4.5 is an example advertisement for an SP. The details of the concepts necessary to define an advertisement template are:

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:service="http://www.cs.cf.ac.uk/conoise/schemas/service#"
    xmlns:profile="http://www.cs.cf.ac.uk/conoise/schemas/profile#"
    xmlns:quality="http://www.cs.cf.ac.uk/conoise/schemas/quality#"
    xmlns:media="http://www.cs.cf.ac.uk/conoise/schemas/media#"
    xmlns:advertisement="http://www.cs.cf.ac.uk/conoise/schemas/advertisement#"
    xml:base="http://www.cs.cf.ac.uk/conoise/samples/SP1advert#">

<advertisement:Advertisement rdf:about="SP1">
    <advertisement:productAdvertised rdf:resource="http://www.cs.cf.ac.uk/conoise/samples/SP1advert#SP1MovieService" />
    <advertisement:productAdvertised rdf:resource="http://www.cs.cf.ac.uk/conoise/samples/SP1advert#SP1NewsService" />
</advertisement:Advertisement>

<media:VideoContent rdf:about="http://www.cs.cf.ac.uk/conoise/samples/SP1advert#SP1MovieService">

    <profile:serviceName>Service Provider 1 Movie Service</profile:serviceName>
    <profile:textDescription>This service provides movies on demand</profile:textDescription>
    <profile:fipaName>SP1@protm:15551/JADE</profile:fipaName>
    <profile:fipaAddress>http://protm.cs.cf.ac.uk:15552/acc</profile:fipaAddress>

    <media:mediaStyle rdf:resource="media:scienceFiction" />
    <media:frameRate rdf:resource="media:24fps" />

    <quality:availability rdf:resource="quality:7DaysAweek" />
    <quality:reliability rdf:resource="quality:99Percent" />

</media:VideoContent>

<media:HTMLContent rdf:about="http://www.cs.cf.ac.uk/conoise/samples/SP1advert#SP1NewsService">

    <profile:serviceName>Service Provider 1 News Service</profile:serviceName>
    <profile:textDescription>This service provides news in HTML</profile:textDescription>
    <profile:fipaName>SP1@protm:15551/JADE</profile:fipaName>
    <profile:fipaAddress>http://protm.cs.cardiff.ac.uk:15552/acc</profile:fipaAddress>

    <media:mediaStyle rdf:resource="media:news" />

    <quality:availability rdf:resource="quality:7DaysAweek" />

</media:HTMLContent>
</rdf:RDF>
```

Schemas

Advertisement element

Service Description elements (MovieService)

Service Description elements (NewsService)

Figure 4.5: Service Advertisement Template

- Schemas: The start of any advertisement consists of a collection of references to the schemas used to define the various elements in the service advertisement. These references include a service profile, a domain specific and a quality schema.

- Advertisement element: This is the first element of the advertisement template. Within this element an SP lists one or more services on offer. It is required that each service offer, is identified by a unique name. A common approach to specifying this unique name is by using URIs as shown in Figure 4.5, e.g., `http://www.cs.cf.ac.uk/conoise/samples/SP1advert#SP1MovieService`.

- Service Description elements: Each service offered within the advertisement element is described in detail using a sequence of statements. This sequence constitutes a service's description. The top element of a service description is the service class name from the domain specific schema, e.g., `Video Content` from the media service schema. The service description includes three important parts:

  - service profile elements such as service name, text description, fipa name and fipa address,

  - any service specific functional attributes, e.g., media style of type "science fiction", and

  - QoS guarantees the SP is prepared to offer with this service. The QoS information can include reference to service specific QoS attributes (e.g., `frameRate`) and service independent QoS attributes (e.g., `availability`).

Once a service advertisement is created it is sent to the YP agent for registration. It is important to note that the representation units of the QoS attribute values within the advertisement template are not specified in the advertisement although, some are represented with a numerical value such as a `frameRate` of 24fps and some are represented using a unit of days (e.g., availability). It is assumed that such domain dependent measurement units are predefined within service and quality schemas. This means these schemas must hold definitions of the possible units of representation for attributes. Thus, based on the schemas, the YP agent can interpret service advertisements.

Service Request

Creating a service request for an SR involves the use of the service and quality schemas. An example service request is shown in Figure 4.6. The request template consists of:

- the request element that allows the YP agent to identify and process the request appropriately, and

- the service requirement.

Each of these elements consists of a sequence of statements that state the capability requirements for a service. The top element of the **service requirement** is the service class name from the domain specific schema, e.g., **Video Content** from the media service schema. The capability requirements include two parts:

- requirements for service specific functional attributes, e.g., media style of type science fiction, and

• quality requirements for a given service request, e.g., reliability of 90 percent.

```
                    <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
                    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
        Schemas     xmlns:quality="http://www.cs.cf.ac.uk/conoise/schemas/quality#"
                    xmlns:media="http://www.cs.cf.ac.uk/conoise/schemas/media#"
                    xmlns:request="http://www.cs.cf.ac.uk/conoise/schemas/request#"
                    xml:base="http://www.cs.cf.ac.uk/conoise/samples/SR1request#">

                    <request:Request rdf:about="SR1">
        Request      <request:productRequested rdf:resource="http://www.cs.cf.ac.uk/conoise/samples/SR1request#MovieService" />
        element       <request:productRequested rdf:resource="http://www.cs.cf.ac.uk/conoise/samples/SR1request#NewsService" />
                    </request:Request>

                    <media:VideoContent rdf:about="http://www.cs.cf.ac.uk/conoise/samples/SR1request#MovieService">
        Service
        Requirement     <media:mediaStyle rdf:resource="media:scienceFiction" />
        elements        <media:frameRate rdf:resource="media:24fps" />
        (MovieService)
                        <quality:availability rdf:resource="quality:7DaysAweek" />
                        <quality:reliability rdf:resource="quality:90Percent" />

                    </media:VideoContent>

        Service     <media:HTMLContent rdf:about="http://www.cs.cf.ac.uk/conoise/samples/SR1request#NewsService">
        Requirement
        elements        <media:mediaStyle rdf:resource="media:news" />
        (NewsService)
                        <quality:availability rdf:resource="quality:7DaysAweek" />

                    </media:HTMLContent>
                    </rdf:RDF>
```

Figure 4.6: Service request with QoS

It is important to note that the QoS attribute value in a service request message represents, the threshold value an SR is prepared to accept for this property (minimum value for positive and maximum value for negative type attributes). Any QoS attribute value in the service advertisement that is either equal or higher (lower in the case of negative attribute type) than that of the service request is treated as meeting the SR's needs. Finally, once created, service requests are sent to the YP agent for matchmaking.

4.3 Matchmaking

Matching a service request with advertisements is performed by the YP agent. In this section, we explain how this is done. Suppose we have a set of advertised services A and a service request R, respectively, where

$$A = \{s_1^a, s_2^a, \ldots, s_n^a\} \qquad R = \{s_1^r, s_2^r, \ldots, s_k^r\}$$

where each s_i^a ($1 \leq i \leq n$) is an advertised service and each s_j^r ($1 \leq j \leq k$) is a requested service. Note that a single service request may ask for several services.

For matchmaking purposes, we assume that an advertised service (s^a) and a requested service (s^r) are represented as follows:

$$s^a = \langle sn, sp, fs, qs \rangle \qquad s^r = \langle sn, fs, qs \rangle$$

where sn is the service name, sp is the service provider, fs is the set of functional specifications and qs is the set of QoS specifications. We refer to these components using the "." notation, that is, $s^a.sn$ refers to the service name of s^a.

Our matchmaking task is to find a set of SPs who offer services that will match any subset of R. That is, we search for

$$M = \{s_i^a \mid s_i^a \succeq R' \subseteq R, 1 \leq i \leq n\}$$

where $s_i^a \succeq R'$ denotes that s_i^a provides a service that satisfies the functional and QoS requirements of each $s^r \in R'$. The following steps describe how M is found.

1. *Determine which advertisements are relevant to R.* This is not simply a process of comparing $s_i^a.sn$ to $s_j^r.sn$. Since our service schema organises classes

of services as a hierarchy, it is necessary to traverse the service schema hierarchy. For example, the schema given in Figure 4.3 defines **VideoContent** and **HTMLContent** as two sub-classes of **Multimedia**. If an SP advertises it offers a **Multimedia** service, then this SP is considered to offer *both* **VideoContent** and **HTMLContent** services together. Thus, to search for SPs who can provide **VideoContent** and **HTMLContent** services, it is necessary to consider advertisements that offer **Multimedia** services too. The following procedure explains how this is performed by the YP component.

input $A = \{s_1^a, s_2^a, \ldots, s_n^a\}$, $R = \{s_1^r, s_2^r, \ldots, s_k^r\}$, ont = service schema

output $Rel \subseteq A$

1 $Rel = \emptyset$

2 for each s_j^r $(1 \leq j \leq k)$

3      $Rel = Rel \cup \{s_i^a.sp \mid s_i^a.sn = s_j^r.sn, 1 \leq i \leq n\}$

4      $p = s_j^r$

5   while $(p \neq null)$

6          $p = getParentClass(p, ont)$

7          $S = \{s_i^a \mid s_i^a.sn = p, 1 \leq i \leq n\}$

8   for each $s \in S$

9              $c = getComponentServices(s)$

10             $Rel = Rel \cup \{s_i^a.sp \mid c \subseteq R, s_i^a.sn = p\}$

11 return Rel

The YP component will first find all the advertisements that have the same service names as those requested (line 3). Then, the YP component recursively traverses up the service schema hierarchy (line 6) to find those services that are more general than s_j^r, but contain component services that are requested in R (lines 9 & 10) and do not contain other non-requested services. This "no more than required" restriction is necessary because currently we assume that an advertised service must be taken in its entirety. For example, one is not allowed to take the HTMLContent service alone from an SP if it has advertised to offer Multimedia services which are not needed. Clearly, SPs who offer more services than necessary (and perhaps therefore will charge more) are undesirable.

2. *Determine which advertisements meet functional requirements.* In this step, the functional requirements specified in each $s^r \in R$ are used to determine which advertised services that the YP component discovered in Step 1 must not be returned to the SR. That is, the YP component performs the following algorithm:

input $Rel = \{s_1^a, s_2^a, \ldots, s_m^a\}$, $R = \{s_1^r, s_2^r, \ldots, s_k^r\}$

output $RF \subseteq Rel$

1 $RF = \emptyset$

2 for each s_j^r $(1 \leq j \leq k)$

3      $RF = RF \cup \{s_i^a.sp \mid s_i^a.sn = s_j^r.sn, s_i^a.fs \geq s_j^r.fs, 1 \leq i \leq m\}$

4 return RF

where $s_i^a.fs \geq s_j^r.fs$ expresses that advertised functionality $(s_i^a.fs)$ must be equal to or better than the requested $(s_j^r.fs)$. For example, if the SR requires **Media style = science fiction**, then movie service advertisements that offer **cartoons** will be disregarded at this stage. This is a fairly straightforward process. If no advertisements can be found, the YP component will send a **failed** message to the SR.

3. *Determine which advertisements meet QoS requirements.* This is similar to Step 2, except that the conditions for matching are different. Assume that $qa \in s_i^a.qs$ is one of the advertised qualities for s_i^a and $qr \in s_j^r.qs$ is one of the requested qualities for s_j^r. The YP component will match qa with qr according to:

advertised (qa)	requested (qr)	matching condition
specified	specified	$qa \geq qr$
specified	unspecified	matching
unspecified	specified	matching

That is, if either qa or qr is unspecified, then the YP agent considers the two qualities unconditionally matching. This is justified because they represent the cases where either the SR is not interested in some QoS property or situations where it is uncertain whether its QoS requirements can be met or not and this cannot be verified. At the end of this step, any advertisements that do not meet the required QoS properties will be dropped from the *RF*, and the details of SPs for the remaining advertisements are returned to the SR.

4.4 Summary

In this chapter we presented an approach to incorporating QoS specifications into service discovery. In our approach we created an extended service description, to allow SPs and SRs to advertise and request services with both functional and QoS requirements. We also described a matchmaking algorithm that allows SRs to find requested services. It is important to note that the presented service and quality classes in this chapter are not fixed schemas. They can be created and modified as required, without affecting the underlying service discovery mechanism. Thus, our approach provides an extensible, dynamic service discovery in an open, distributed computing environment.

This chapter explained how the YP component performs matchmaking between advertised and requested services during the service discovery step of our QoS model. As our model also has a second step, the service selection step, the SR may use the QoS component in the selection step to establish what can really be expected from the SPs returned by the YP component. This is particularly useful in cases where some SPs cannot be trusted or some QoS requirements specified by the SR are unspecified by the SPs. The QoS component can in such cases help to establish some "facts" about the "unknowns", based on other users' experience with the services. In the next chapter we will present our solution to QoS assessment and selection.

Chapter 5

Incorporating QoS in Service Selection

5.1 Introduction

In this chapter, we consider the problem of incorporating quality of service (QoS) assessment in the service selection. This is an important problem to consider because, just like in any other business environment, it is possible that SPs may not provide what they promise. It is essential, therefore, that an SR agent should select a service that not only meets the quality preference requirements based on the SP's service advertisement, but also one that has a proven track record.

The ability to perform service selection is critical if we are to realise the potential of the SOC paradigm. Many techniques have been proposed for calculating the quality of a service, and they do so typically by collecting quality ratings from the users of the service, then combining them in some way to derive the quality of the service. We

argue that collecting quality ratings alone from the users is not sufficient for deriving a reliable and accurate quality measure for a service. This is because different users often have different expectations on the quality of a service and their ratings tend to be closely related to these expectations, i.e., how their expectations are met. In this chapter, we describe our quality of service assessment approach which incorporates user expectations. That is, we collect expectations as well as ratings from the users of a service, then calculate the quality of the service using only the ratings that have similar expectations.

To illustrate this, consider an example based on the multimedia scenario introduced in Chapter 3. Suppose that we have three SPs (SP1, SP2 , SP3) who offer a multimedia movie service to PDA or mobile phone users. Suppose also that there are six SRs (SR1, .. , SR6) who have used the services, and each of them has been asked to rate the quality of the service he or she has received in terms of movie frame rate. Table 5.1 shows the quality ratings collected from the six SRs, where ratings are expressed as real numbers in $[0, 1]$ with 0 representing the most unsatisfactory quality and 1 the most satisfactory. For simplicity of presentation, we assume that the aggregate quality rating for each SP is derived by combining the individual ratings using a simple arithmetic average. So according to Table 5.1, SP2 offers the best service with respect to movie frame rate.

While various methods may be employed to aggregate the collected ratings more rationally, for example, using a weighted average so that the reputation or trust of the user may be taken into account [127, 208], this approach to quality rating calculation suffers from two fundamental weaknesses:

Table 5.1: Collected Quality Ratings

SRs	SP1 frame rate	SP2 frame rate	SP3 frame rate
SR1	0.3		0.3
SR2	0.8	0.9	
SR3	0.3		1.0
SR4		0.8	
SR5	0.5		0.1
SR6	0.6	0.3	
Aggregate rating	0.50	0.67	0.47

- First, users are invited to rate a service in "absolute" terms, e.g., 0.3 or 0.8 out of 1.0 in our example. Such quality ratings may not be very meaningful or can even be misleading in some cases, because the context within which the ratings are derived is not known. For example, SR1 rated SP1 low perhaps because SP1's movie frame rate was not fast enough for her, but this does not necessarily mean that the same frame rate is not good enough quality for a different SR, e.g., SR2.

- Second, the aggregate quality rating for a service is derived "statically" using all the ratings collected from the SRs. This does not take into account the fact that some of the ratings may not be relevant to a particular quality assessment request. For example, if the current request was to assess the quality of SP1, SP2 and SP3 in terms of their ability to stream movies at 16 frames per second (fps), then SR6's rating should not be included in the quality calculation, since SR6 had expected a minimum of 30 fps from SP2 and clearly SR6 would not be satisfied with frame rate of 16.

In this chapter, we address these two problems by introducing a new model for collecting and calculating QoS "relatively". That is, we attempt to collect from service users QoS ratings as well as their expectations on QoS, so that we can measure QoS in relative terms, i.e., how well a delivered service meets the user's expectations. Based on user expectations, we also propose to calculate the quality of a service dynamically at the time a request for QoS assessment is made, and use only the ratings that have similar expectations.

The rest of the chapter is organised as follows. Section 5.2 describes representation of QoS expectations for the purpose of quality assessment in an SOC environment. Section 5.3 discusses how user expectation is collected and QoS rating calculated by the QoS agent. We give an example to show that our approach results in a more accurate and meaningful measure for quality of service. Finally, Section 5.4 summarises this chapter.

5.2 Expectation based Quality of Service Assessment

To support dynamic and automated service selection, it is essential to understand what *service quality assessment* actually entails. As discussed in Chapter 2, different views exist in different studies and there are different perspectives of what it is [36, 86, 103, 124, 139, 188]. The following three views are, however, most common:

- *Quality as Functionality.* This view considers quality assessment in terms of the amount of functionality that a service can offer to its users. For example, if an

SP (e.g., SP1) allows you to select different positions of cameras from which you may watch a football game, and if this functionality is not provided by other SPs (e.g., SP2 or SP3), then SP1 can be considered as offering a better quality than SP2 and SP3 do if this feature is required.

- *Quality as Conformance.* This view sees quality assessment as being synonymous with meeting the specifications given. For example, if SP1 specified in its service agreement that it would provide 1 Mb/s bandwidth for its movie service and SP1 did provide users with 1 Mb/s bandwidth (or more) at all times in its operation, then SP1 is usually considered as offering good quality of service.

- *Quality as Reputation.* This view links quality assessment to a user's perception of a service in general. It is worth noting that this perception is typically built over the time of the service's existence. For example, a well-established movie service may be considered as offering good quality to its users by the public in general, due to its reputation built over many years as a movie service provider. In other words, this view recognises quality as an innate excellence and reflects the belief that though style and taste may change, there is something enduring about a high quality service.

These different views of quality require QoS to be measured differently depending on the view required. Quality as functionality characterises the design of a service and can only be measured by comparing the service against other services that offer similar functionality. Quality as conformance, on the other hand, can be measured for each service individually, but requires the actual experience of using the service in order

to measure the delivery against a set standard. Finally, reputation can be regarded as a reference to a service's consistency over time in offering both functionality and conformance qualities, and can therefore be measured through these two types of quality over time.

Having explored these different views of quality, it is worth noting the following two points. First, each view defines some meaningful and plausible aspect of quality, but none of them is in fact a complete measure of it. For example, measuring quality as conformance alone would suggest that two services are of equal quality if they have equal conformance, regardless of whether one might offer more functionality than the other. Such a measure of quality is obviously incomplete. Thus, it is important to understand an assessment of quality in context. Second, the above views define quality, but do not suggest what constitutes a high quality. For example, although functionality is considered as a measure of quality, it does not necessarily imply that the more the functionality, the higher the quality. Whether a particular service offers good quality or not can be highly dependent upon an individual's quality preference and current requirements.

In an SOC environment, it is most relevant to consider how quality as conformance may be monitored and calculated [49]. In this thesis, therefore, we adopt the conformance view of QoS. More formally, we define conformance based QoS as follows.

Definition 1 *Let S be a service and $A_1, A_2, \ldots, A_n$ be a set of attributes that describe S and with which we will assess the quality for S. Assume that for each A_i, A_i^a is the advertised quality (or the quality that the SP promised to offer), and A_i^d is the*

delivered quality (or the actual quality that the SP delivered). Then the QoS for A_i is given by

$$Q_{A_i} = f(A_i^a, A_i^d)$$

where f is a function that calculates the conformance between A_i^a and A_i^d.

The above definition captures the notion of conformance generically, but does not specify how A_i^a and A_i^d may be obtained and Q_{A_i} calculated. In practice, it may not be realistic to expect every A_i to have an A_i^a value specified by the SP, and its A_i^d value monitored by the SR and Q_{A_i} calculated by the system automatically. Often, we need user feedback to help assess the quality of a service.

While the need to involve users in QoS assessment is well recognized, existing methods tend to collect quality ratings (Q_{A_i}) only from the users (as discussed in Chapter 2). This is inadequate if we wish to measure quality as conformance according to Definition 1. In this thesis, we propose to collect "fuller" ratings from the users for QoS assessment.

Definition 2 *Let U be an SR and A_i be an attribute of a service S. A quality rating on A_i by U is a triple*

$$\langle E(A_i), P(A_i), R(A_i) \rangle$$

where $E(A_i)$ represents the quality that U expects from A_i, $P(A_i)$ the actual quality of A_i perceived or experienced by U after using S, and $R(A_i)$ the quality rating that U gives to A_i.

Collecting $\langle E(A_i), P(A_i), R(A_i) \rangle$ from the SR can perhaps be considered as a way of materialising the conformance calculation function introduced in Definition 1.

Instead of relying completely on the system for monitoring A_i^d and calculating Q_{A_i}, we allow $P(A_i)$ and $R(A_i)$ to be obtained from the SR too. The use of $E(A_i)$ represents a shift from using SP advertised values to SR expectations on quality in QoS calculation. This is significant. While some correlation between A_i^a and $E(A_i)$ can be expected - SRs are likely to be influenced by advertising in forming their expectations, it is important to realise expectations are not solely based on advertisement. Other factors, such as the SR's past experience with the service, the price the SR is paying for the service, or the recommendation by a friend for the service can all influence the SR in forming her expectation on the quality of the service. Thus, by including SR expectations as part of the SR rating on a service, we can hope to interpret such ratings more meaningfully.

To explain how the proposed QoS assessment model works, consider the multimedia scenario we introduced in Table 5.1 again. Suppose that we still ask the six users to rate the three movie services in terms of frame rate, *fr*, but this time use the expectation model, we introduced here. Assuming that we represent $E(\text{fr})$, $P(\text{fr})$ and $R(\text{fr})$ all as real numbers in $[0, 1]$, Table 5.2 shows the ratings collected from the users, where each entry represents a $\langle E(\text{fr}), P(\text{fr}), R(\text{fr}) \rangle$ triple.

Table 5.2: Expectation based Quality Ratings

Users	SP1 $\langle E(\text{fr}), P(\text{fr}), R(\text{fr}) \rangle$	SP2 $\langle E(\text{fr}), P(\text{fr}), R(\text{fr}) \rangle$	SP3 $\langle E(\text{fr}), P(\text{fr}), R(\text{fr}) \rangle$
SR1	$< 0.9, 0.7, 0.3 >$		$< 0.7, 0.5, 0.3 >$
SR2	$< 0.4, 0.4, 0.8 >$	$< 0.5, 0.5, 0.9 >$	
SR3	$< 0.8, 0.6, 0.3 >$		$< 0.4, 0.5, 1.0 >$
SR4		$< 0.6, 0.6, 0.8 >$	
SR5	$< 0.9, 0.7, 0.5 >$		$< 0.9, 0.5, 0.1 >$
SR6	$< 0.9, 0.7, 0.6 >$	$< 0.7, 0.5, 0.3 >$	

How a user arrived at a particular rating may never be known to us, but it is interesting to speculate what the ratings shown in Table 5.2 might suggest. The majority of the users of SP1 and SP3 seem to have high expectations (probably as a result of some effective recommendations or advertising effort), but do not seem to get what they expect (perhaps due to the unexpected level of business that SP1 and SP3 have achieved due to these factors). SP2, on the other hand, is the opposite: SRs do not have high expectations but are generally satisfied with the service. In the following section, we show how this difference in expectation is taken into account when assessing QoS for services.

5.3   Collecting and Calculating QoS Ratings

In Section 3.3, the QoS agent was introduced and its role in service selection was discussed. This section discusses the rating collection and calculation functionalities of the QoS agent. The architecture view of the QoS agent is shown in Figure 5.1.
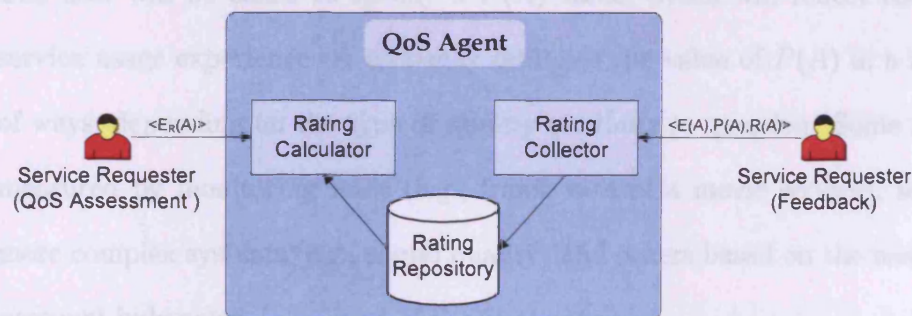

Figure 5.1: The System Architecture of the QoS Agent

5.3.1 The Rating Collector

As discussed in Section 3.5.1, the Rating Collector is responsible for collecting QoS related information from SR agents. It is worth noting that in the previous discussion, we implicitly assumed that the SR agent will notify the Rating Collector of QoS usage information in the format of $E(A)$, $P(A)$ and $R(A)$. There are different methods that may be used to establish $E(A)$, $P(A)$ and $R(A)$ values and these are not covered here, as we assume they are available. This is a realistic assumption as online reviews have become increasingly popular as a way to judge and share the quality of various services [53, 8, 181]. In the following we describe how an SR will establish $E(A)$, $P(A)$ and $R(A)$ using the multimedia scenario.

- The user will be asked to specify an $E(A)$ value, which may reflect the influences caused by the service advertisement, the user's own experience with the service or some recommendation from a third party.

- The user will be asked to specify a $P(A)$ value, which will reflect the user's service usage experience. A user may establish the value of $P(A)$ in a number of ways, depending on the type of quality attribute in question. Some may be measured by monitoring tools (e.g., frame rate of a movie service), some by more complex systems (e.g., sound quality) and others based on the user's own personal judgement (e.g., level of detail of news).

- The user will be asked to specify a $R(A)$ value, which reflects the user's degree of satisfaction with the service. A user may establish the $R(A)$ value from $E(A)$ and $P(A)$. For example, if $E(\text{frame rate}) = 24$ and $P(\text{frame rate}) = 22$, then

R(frame rate) may be calculated using the following formula:

$$R\text{(frame rate)} = min(1, \frac{P\text{(frame rate)}}{E\text{(frame rate)}}) = min(1, \frac{22}{24}) = 0.92$$

5.3.2 The Rating Calculator

As discussed in Section 3.5.3, the Rating Calculator is responsible for calculating the QoS value from the collected ratings. To aggregate individual ratings into an overall assessment of the quality for a given service S, two calculations are necessary:

1. combining individual ratings for each attribute A_i of S into an aggregate rating for A_i, and

2. combining the ratings for individual A_i's into an overall rating for S.

Currently, we treat all quality attributes of a service as of equal importance and the overall rating for S is derived by a simple average of the individual ratings for its attributes. But it is possible to consider a weighted average so that the fact that some attributes are more significant than others may be taken into account [121].

How to combine individual ratings for each A_i into a single aggregate rating, needs some further explanation. In contrast to many existing methods which simply aggregate *all* of the collected ratings on A_i, our approach is to selectively aggregate only the ratings that have similar expectations. That is, we allow a quality assessment request R to specify a quality expectation on A_i, $E_R(A_i)$, and derive an aggregate quality rating for A_i by using only the ratings in the Rating Repository that have similar expectations to $E_R(A_i)$. More specifically,

- If R does not specify any quality expectation on A_i, then $Q(A_i)$, the quality rating for A_i, is

$$Q(A_i) = \sum_{j=1}^{k} w_j \times R_j(A_i)$$

where $R_j(A_i)$ is user j's rating on A_i and w_j is a weight. This is equivalent to the majority of existing approaches to quality calculation, where the overall rating for A_i is a weighted sum of the individual ratings, and the weights are used to allow factors such as reputation or trust to be taken into account [127, 208].

- If R specifies a quality expectation $E_R(A_i) = \alpha \in [0,1]$ on A_i, i.e., the quality expectation on A_i is α, then

$$Q(A_i) = \sum_{j=1}^{m} w_j \times R'_j(A_i)$$

where $R'_j(A_i)$ is the rating element of $\langle E'_j(A), P'_j(A), R'_j(A) \rangle$ in the Rating Repository whose corresponding expectation element $E'_j(A_i)$ is *similar* to $E_R(A_i)$ = α. In this section, we use a range based criterion for determining whether the two are similar: $E'_j(A_i)$ and $E_R(A_i)$ = α are compatible or similar if $|E'_j(A_i) - \alpha| \leq \delta$, where δ is a constant. More complex forms of similarity test are possible, for example, by specifying quality expectations as "ranges" over $[0,1]$ (instead of *points*) and by allowing fuzzy matching between one or more $E'_j(A_i)$s and $E_R(A_i)$s.

By aggregating individual quality ratings dynamically at the time when a QoS assessment request is made and by comparing the raters' and the requester's expectations on qualities, our approach is able to calculate QoS in "context", that is,

to use the ratings which are actually relevant to the context within which the QoS assessment request is made.

Now consider our multimedia example and Table 5.2 again. Suppose that the QoS agent has been asked to assess QoS for all three SPs in terms of frame rate, given $E_R(\text{fr}) = $ unspecified, $E_R(\text{fr}) = 0.5$ and $E_R(\text{fr}) = 0.8$, respectively. Assuming that we have $\delta = 0.1$, the result of the calculation by the QoS agent is given in Table 5.3.

Table 5.3: Calculated Quality Ratings for SPs

Expectation	SP1	SP2	SP3
$E_R(\text{fr}) = $ unspecified	0.50	0.67	0.47
$E_R(\text{fr}) = 0.5$	0.80	0.85	1.00
$E_R(\text{fr}) = 0.8$	0.43	0.30	0.20

As can be seen from Table 5.3, the quality ratings for SPs can vary with respect to expectations. For example, when the expectation is $E_R(\text{fr}) = 0.5$, SP3 emerges as the best service provider, whereas when the expectation is changed to $E_R(\text{fr}) = 0.8$, we have SP1 as the best. This is in contrast to conventional approach to QoS assessment in SOC that do not consider user expectations. This is equivalent to setting $E_R(\text{fr}) = $ unspecified, resulting in SP2 being the best provider for all cases. Thus, our method gives a more meaningful rating for a service on a case-by-case basis. In Chapter 6, we will show through experiments that the approach proposed here can result in a more meaningful measure for quality of service.

5.3.3   Fuzzy Similarity based QoS Assessment

In Section 5.3.2 we presented a range based similarity assessment of expectations. However, there is a weakness associated with this approach to QoS assessment. QoS attributes may display some relationships among the attributes. For example, a high value on **security** is often linked to a lower **delivery time**, and a higher value of **completeness** may be linked with a low **preciseness** value. Also some attributes are directly linked, for example, higher values of **framerate** and **display size** would mean a high value of **bandwidth** too. Similarity assessment needs to consider presence of such relationships so that relevant ratings are selected. For example, consider two QoS attributes, **framerate** and **bandwidth**, that are linked by a direct relationship, and assume that the SR's expectations are $E_R(A_{framerate}) = 24fps$ and $E_R(A_{bandwidth}) = 100kbps$. This would result in the rating with $E'_j(framerate) = 24fps$ and $E'_j(bandwidth) = 2000kbps$, to be used in assessment, as $E'_j(framerate) = 24fps$ would match the expectation $E_R(A_{framerate}) = 24fps$ perfectly. Here, even though the two attributes are related (i.e., a change in either of the attributes can affect the other), only the rating of one attribute (i.e., $R'_j(framerate)$) is selected. Therefore, such an expectation matching using a single attribute would be inaccurate.

We propose to address this issue by introducing a fuzzy based [101] similarity assessment that calculates similarity for a group of "related" QoS attributes. In the following we describe how a fuzzy based assessment can be made and compare this to the range based approach.

## QoS Rating Calculation based on Fuzzy Assessment

As QoS attributes can have some relationships among them and can influence each other, we introduce the concept of QoS attribute group which is a set of related attributes. A QoS attribute group is: $G = \{A_1, A_2, \ldots, A_n\}$, where $A_1, \ldots, A_n$ are the related QoS attributes. The aggregated quality rating $Q(G)$ for G is derived by a slightly modified formula from the one described in Section 5.3.2. More specifically, if an *SR* specifies a quality expectation $E_R(G) = \{E_R(A_1), E_R(A_2), \ldots, E_R(A_n)\}$

$$Q(G) = \frac{\sum_{j=1}^{m} SimScore \times R'_j(G)}{\sum_{j=1}^{m} SimScore} \tag{5.1}$$

where $R'_j(G)$ is an average of user j's ratings on the attributes of G and *SimScore* is the corresponding similarity score, indicating "how similar" $E'_j(G)$ is to $E_R(G)$. $E'_j(G)$ is user j's quality expectations on the attributes of G. As part of the quality assessment, a threshold value can be set for *SimScore* to include only the corresponding $R'_j(G)$ that either equals or exceeds the threshold value. In the following we describe how a simple fuzzy based similarity assessment can be used to determine the similarity score (*SimScore*). For ease of presentation, we use an example QoS attribute group to describe the fuzzy assessment approach based on the multimedia scenario.

Fuzzy Representation for Similarity Assessment

Suppose that we have a set of QoS attribute groups, $Z = \{G_1, G_2, \ldots, G_n\}$. A QoS attribute group, `performance(pf)`, for example, may consists of attributes `response time(rt)` and `successful completion(sc)`. That is:
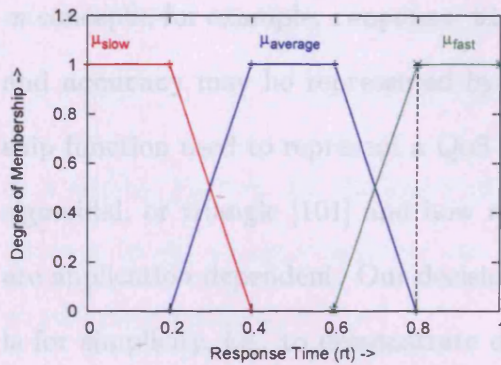
Figure 5.2: Membership functions for rt
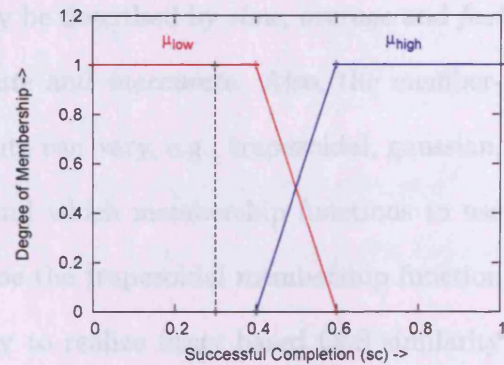
Figure 5.3: Membership functions for sc

$$G_{pf} = \{A_{rt}, A_{sc}\} \tag{5.2}$$

The expectation on each attribute is mapped to a membership value (or degree of membership) in $[0, 1]$ using a fuzzy membership function [96]. The membership functions of rt and sc, for example, are shown in Figures 5.2 and 5.3 respectively. Both are trapezoidal functions. As can be seen from Figures 5.2 and 5.3, the fuzzy sets for each attribute can be described as follows:

$$A_{rt} = \left\{ slow, avg, fast \right\}$$

and,

$$A_{sc} = \left\{ low, high \right\}$$

For example, an instance of response time $= 0.8$ will map to the fuzzy concept of high with a 100% membership. Note that each QoS attribute may be described by

m concepts, for example, **response time** may be described by *slow, average* and *fast* and **accuracy** may be represented by *accurate* and *inaccurate*. Also, the membership function used to represent a QoS attribute can vary, e.g., trapezoidal, gaussian, sigmoidal, or triangle [101] and how many and which membership functions to use are application dependent. Our decision to use the trapezoidal membership function is for simplicity, i.e., to demonstrate one way to realize fuzzy based QoS similarity assessment. It is possible to use other functions.

Using the above discussion on membership functions, we can formulate a vector fuzzy description of each QoS attribute group as:

$$
G = \begin{pmatrix}
\mu_{11}(A_1), & \mu_{12}(A_1), & \cdots & \mu_{1m(1)}(A_1), \\
\mu_{21}(A_2), & \mu_{22}(A_2), & \cdots & \mu_{2m(2)}(A_2), \\
\cdots, & \cdots, & \cdots, & \cdots, \\
\mu_{n1}(A_n), & \mu_{n2}(A_n) & \cdots & \mu_{nm(n)}(A_n),
\end{pmatrix}
\tag{5.3}
$$

where, $\mu_1, \mu_2, \ldots, \mu_m$ represent the fuzzy concepts that describe attribute A. Note that, a default value zero is used for any missing values in the vector. Using Equation 5.3 we can now describe the QoS attribute group **performance** (G_{pf}), as follows, assuming that we have $E_R(G_{pf}) = \{0.8, 0.3\}$ and the membership functions are as given in Figures 5.2 and 5.3:

$$
E_R(G_{pf}) = \begin{pmatrix}
\mu_{slow}(x, 0.8), & \mu_{average}(x, 0.8), & \mu_{fast}(x, 0.8) \\
\mu_{low}(x, 0.3), & \mu_{high}(x, 0.3), & 0
\end{pmatrix}
\tag{5.4}
$$

which is then:

$$E_R(G_{pf}) = \begin{pmatrix} 0, & 0, & 1 \\ 0, & 1, & 0 \end{pmatrix} \tag{5.5}$$

In a similar manner, all the expectation values of the corresponding groups of attributes in the rating repository are mapped to the fuzzy space using the defined membership functions.

Similarity Score Calculation Using Fuzzy Similarity Assessment

The proposed approach to calculating similarity score is based on using fuzzy similarity relations [213], which represent the degree of similarity between two fuzzy values from the same universal set U. The similarity score between two fuzzy sets for an attribute group is identified as:

$$SimScore = 1 - \frac{|(E'_j(G)) - (E_R(G))|}{\max((E'_j(G)), (E_R(G)))} \tag{5.6}$$

We use Equation 5.6 in Equation 5.1 to calculate an aggregated quality rating for a group. Now consider the following example. Suppose that the QoS agent has been asked to assess QoS for an SP in terms of performance, given $E_R(G_{pf}) = E_R(\{A_{rt}, A_{sc}\}) = \{0.8, 0.3\}$ and a threshold value ≥ 0.5. Note that, we have mapped $E_R(G_{pf})$ to values in the fuzzy space shown in Equation 5.5. Table 5.4 shows four expectations $E'(A_{rt})$ and $E'(A_{sc})$ of $E'_j(G_{pf})$ in the Rating Repository. Each of these expectations is mapped to the fuzzy space in a similar manner and the result is given in Table 5.5. Finally, Equation 5.6 is used to calculate *SimScore*. For example, the *SimScore* for SR2 is calculate by:

$$SimScore = 1 - \frac{((|0 - 0|) + (|0.5 - 0|) + (|0.5 - 1.0|) + (|1.0 - 1.0|) + (|0 - 0|))}{(0 + 0.5 + 1.0 + 1.0 + 0)}$$

$$= 1 - \frac{(0 + 0.5 + 0.5 + 0 + 0)}{2.5}$$

$$= 1 - \frac{1.0}{2.5}$$

$$= 0.6$$

Table 5.4: Expectations in the Rating Repository

SRs	$E'(A_{rt})$	$E'(A_{sc})$
SR1	0.9	0.2
SR2	0.7	0.4
SR3	0.8	0.9
SR4	0.7	0.9

Table 5.5: Mapping expectations to the fuzzy space

SRs	μ for A_{rt}			μ for A_{sc}	
	Slow	Average	Fast	Low	High
SR1	0	0	1.0	1.0	0
SR2	0	0.5	0.5	1.0	0
SR3	0	0	1.0	0	1.0
SR4	0	0.5	0.5	0	1.0

SimScores for all the SRs are similarly calculated for the given $E_R(G_{pf})$ and are shown in Table 5.6 [1]. Table 5.6 also shows the perceptions (P') and ratings (R') for A_{rt} and A_{sc}, and the aggregated ratings of the attribute group $R'(G_{pf})$ for all the SRs. Based on the given threshold value of the *SimScore*, only the ratings of SR1 and SR2 are selected to calculate the aggregated quality rating for the attribute group G_{pf}. This demonstrates that the fuzzy quality assessment is capable of grouping related attributes together and selecting similar expectations.

Table 5.7 shows the aggregated results of the quality assessment for G_{pf} based on range-based assessment. Assuming that we have $\delta = 0.1$ for range-based assessment, it can be seen from Table 5.4 that all the ratings have "similar" expectation values to the requester's expectation on **response time** ($0.7 \leq E'(A_{rt}) \leq 0.9$). Using the

---

[1] $\sqrt{}$ represents the *SimScore* greater than or equal to the threshold value and * represents the ratings in the rating repository selected for QoS assessment

Table 5.6: Fuzzy-based quality assessment

SRs	A_{rt}		A_{sc}		$R'(G_{pf})$	$SimScore$	$Q(G_{pf})$
	$P'(A_{rt})$	$R'(A_{rt})$	$P'(A_{sc})$	$R'(A_{sc})$			
SR1	0.8	0.9*	0.2	1.0*	0.95	1.0$\sqrt{}$	
SR2	0.7	1.0*	0.3	0.8*	0.9	0.6$\sqrt{}$	0.931
SR3	0.3	0.2	0.2	0.1	0.15	0.33	
SR4	0.1	0.1	0.2	0.1	0.1	0.14	

range based assessment, all the ratings of A_{rt} are selected to provide an aggregated quality rating, $Q(A_{rt}) = 0.55$, as shown in Table 5.7 [2]). Here, it is interesting to observe that the perceptions and rating on A_{rt} from SR3 and SR4 are very low, even though they have similar expectations to SR1 and SR2 (see Table 5.4). The reason for this may be due to a lower than expected quality of A_{rt} for those SRs, whereas SR1 and SR2 received these as expected. The relationship between A_{rt} and A_{sc} may also have caused the fall in perception on A_{sc} to reflect the fall in the A_{rt}. The range-based assessment is not able to capture such effects, and thus penalizes a service provider by providing a lower $Q(G_{pf})$ even though the service provider is able to generally meet the expectation of $E_R(\{A_{rt}, A_{sc}\}) = \{0.8, 0.3\}$ (see Table 5.6 and 5.7). A fuzzy based assessment, on the other hand, does not suffer from this problem as which ratings to select from the rating repository is based on the similarity distance between the set of related attributes. This gives a more meaningful QoS assessment for services when relationships exist between multiple attributes.

---

[2]* represents the ratings in the rating repository selected for QoS assessment

Table 5.7: Range-based quality assessment

SRs	A_{rt}		A_{sc}		$Q(G_{pf})$
	$P'(A_{rt})$	$R'(A_{rt})$	$P'(A_{sc})$	$R'(A_{sc})$	
SR1	0.8	0.9*	0.2	1.0*	
SR2	0.7	1.0*	0.3	0.8*	
SR3	0.3	0.2*	0.2	0.1	
SR4	0.1	0.1*	0.2	0.1	
$Q(A_i)$		0.55		0.9	0.725

5.4 Summary

In this chapter we presented a novel expectation based model for calculating QoS in an SOC environment, that met part of our research aims specified in Section 1.1. More specifically, it enabled consumers to assess the quality of a service before a decision to use it is made, by allowing the QoS agent to determine the QoS rating for the service. This model is founded on the following basic observation: if A rates the quality of S as x, then this rating is only meaningful to B if A and B have similar expectations for S. So instead of aggregating all the quality ratings collected from users of services, we propose to calculate QoS dynamically at the time a QoS assessment request is made, and use only the ratings where similar expectations to that of the request have been noted. This can lead to more accurate and meaningful rating of QoS assessment. Fuzzy based approach was used to demonstrate the case where multiple attributes are related. The choice to use the fuzzy based approach instead of range based approach was because it aligned well with the requirement to map multiple attributes that are related as a group. The difference in QoS ratings produced using these QoS assessment methods was evaluated by comparing the accuracy of the ratings. This evaluation concluded that the fuzzy based approach would make

a better candidate for QoS assessment in presence of multiple attributes that may be related.

In the next chapter we present an empirical evaluation of our proposed expectation based QoS assessment model, and verify our claim that incorporating expectations into QoS assessment can result in a more meaningful measure.

Chapter 6

Evaluation and Results

6.1 Introduction

Chapter 5 discussed the service selection process that uses the expectation based assessment algorithms. To show that this approach can result in a more meaningful measure for QoS, we conducted two sets of experiments. The first set was designed to show that the expectation based model allows QoS ratings to be dynamically generated by matching expectations from historical QoS data with that of SRs expectations. As a result, a personalized QoS assessment is established for the SR. The second set of experiments was to show that the selection process is able to accurately identify and adapt to changes in delivered QoS. For example, we demonstrate services that provide constant QoS get selected often, while services, whose delivery of QoS deteriorates, are rejected often in the selection process.

6.2 Experiment 1: Personalized QoS Assessment

A range of simulations were carried out for these experiments, and we used synthetically generated QoS rating data throughout. In the following, we will first describe how the QoS data is generated, then analyse the experimental results and compare the outcomes with the non-expectation based approach to QoS assessment's results.

6.2.1 QoS Data Generation

Our experimental QoS rating data is generated in three steps. First, a set of values is generated to represent the actual QoS delivered by an SP w.r.t. an attribute A over a period of time. We call this set of data the *delivered QoS*. We express a set of delivered QoS data as real numbers in $[0, 1]$ which represent the available quality range. A set of delivered QoS data can be generated to have specific "patterns", so that they simulate some real world scenarios. For example, the set of data shown in Figure 6.1 exhibits a near constant QoS level (around 0.5), indicating a fairly stable delivery of service.

Next, we generate expectations, or the values for $E(A)$, for the given set of delivered QoS data. We generate three groups of expectations: (1) low expectations (which are below the delivered QoS), (2) average expectations (which are largely at the same level as the delivered QoS), and (3) high expectations (which are above the delivered QoS). For example, for the delivered QoS data given in Figure 6.1, we generate three sets of expectations around $E(A) = 0.2$, $E(A) = 0.6$ and $E(A) = 0.9$, as shown in Figure 6.2.
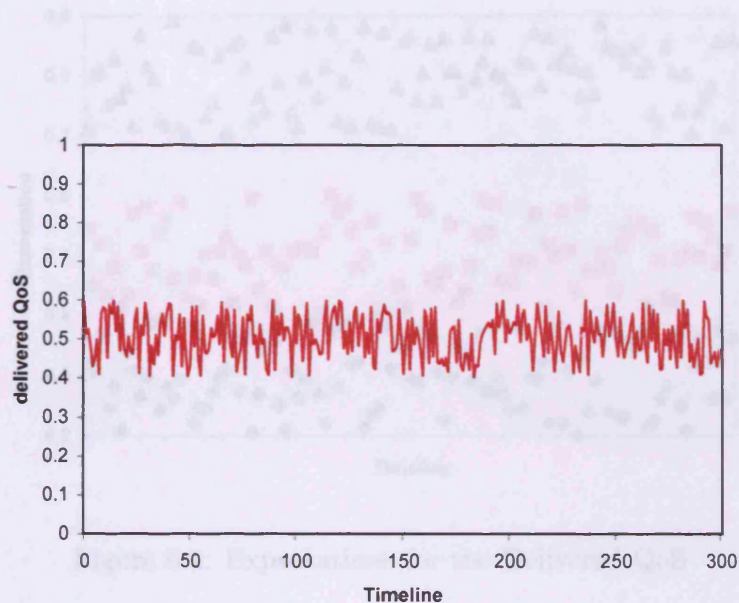
Figure 6.1: A Near Constant Delivery of QoS

Again, we express expectations as Real numbers in $[0, 1]$. Each generated expectation is linked to a single item in the delivered QoS data set. Note that we allow a range for variation in the generation of all three groups of expectations. This is to mirror the real world situation where people may have broadly similar expectations on a quality, but the individual expectations they have can still vary.

Finally, we generate ratings, or the values for $R(A)$. We assume that delivered QoS is accurately observed by the users. That is, the values for $P(A)$ in our experiments are assumed to be the same as the delivered QoS. The ratings are then generated by using a function that takes an expectation and an associated item in the delivered QoS data set as its input, and performs the following calculation:
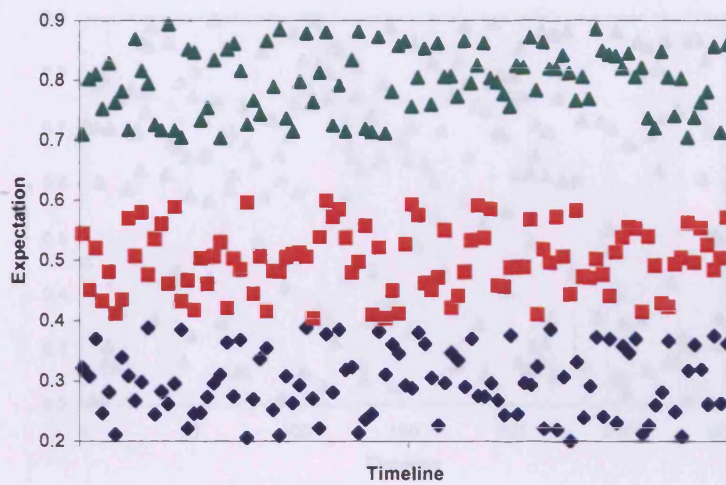
Figure 6.2: Expectations for the Delivered QoS

$rating \in [0.0, 0.4]$ if $P(A) < E(A) - \delta$, where δ is a constant

$rating \in [0.4, 0.6]$ if $E(A) - \delta \leq P(A) < E(A)$

$rating \in [0.6, 1.0]$ if $P(A) \geq E(A)$

For example, for the delivered QoS data set given in Figure 6.1 and the expectations in Figure 6.2, we generate a set of ratings shown in Figure 6.3 ($\delta = 0.1$). It is worth noting that without the expectation information, the distribution of the ratings appears to be rather random in Figure 6.3 and it is hard to see why a particular rating should be high or low. Yet a non-expectation based method will aggregate *all* the ratings in deriving an overall QoS rating, without taking this into account.
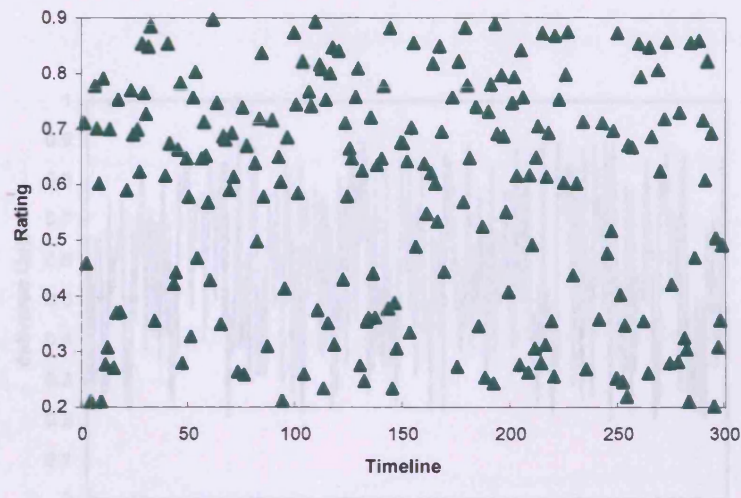
Figure 6.3: Ratings Derived from the Delivered QoS and Expectations

6.2.2 Analysis of Results

We now describe the tests we performed and analyse the results. In addition to the near constant delivered QoS data set we introduced in Figure 6.1, we have also tested our method on two other cases representing fluctuating and gradually falling delivered QoS. These cases are shown in Figures 6.4 and 6.5. These three cases allowed us to study the effectiveness and robustness of our proposed expectation based QoS calculation method in some realistic settings.

Effectiveness of QoS Assessment

In the first set of experiments, we tested the effectiveness of our QoS assessment method on all three cases. Figure 6.6 shows the results of running our expectation based method on the near constant case, with $E_R(A)$ set to 0.2, 0.6 and 0.9, respectively. For reference and comparison, we have also run a non-expectation based QoS

calculation method (which simply averages all the content on the same set of data,
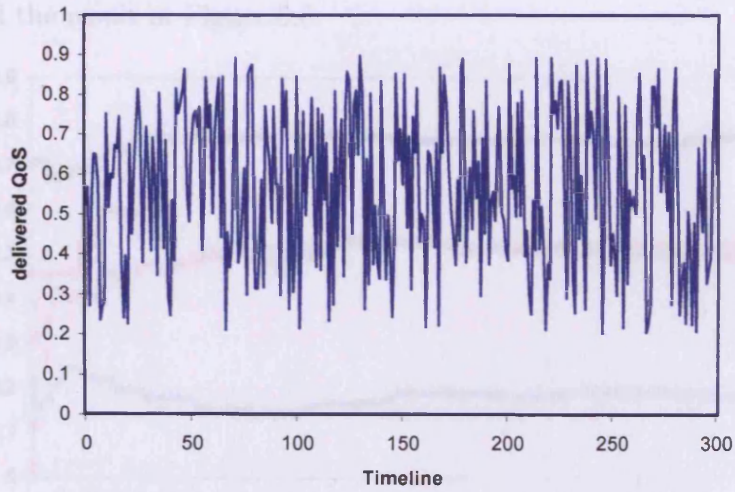
and included the

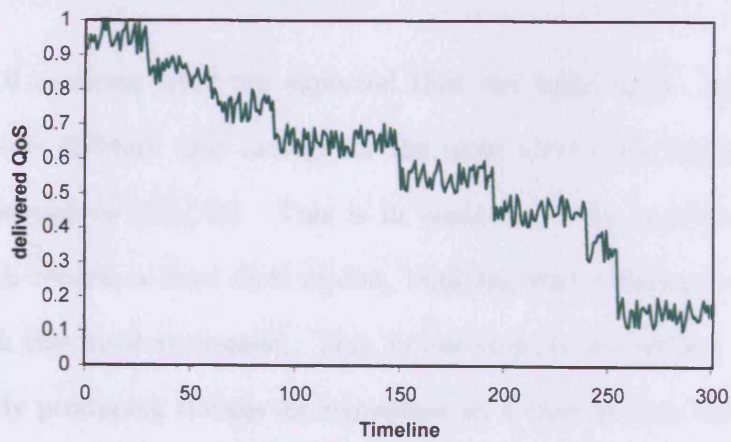Figure 6.4: A Fluctuating Delivery of QoS

Figure 6.5: A Falling Delivery of QoS

calculation method (which simply averages all the ratings) on the same set of data, and included the result in Figure 6.6.
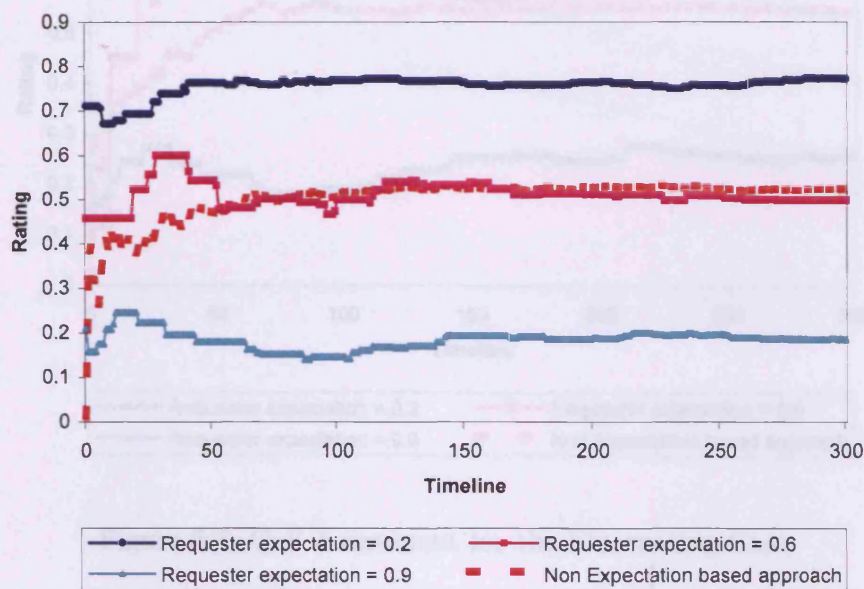

Figure 6.6: QoS Assessment for the Near Constant Case

Figure 6.6 confirms what we expected that our expectation based assessment method delivers different QoS ratings for the same service for requesters who have different expectations $(E_R(A))$. This is in contrast to the non-expectation based method which returns a fixed QoS verdict, implying that different requesters would all agree with this fixed assessment. This, in our view, is not entirely realistic in the real world. By producing ratings for requesters on a case by case basis, our method is more likely to produce ratings that are meaningful to and will be agreed by the individual requesters. Figure 6.7 shows the same runs of our experiments, but using the fluctuating delivered QoS data set as input.
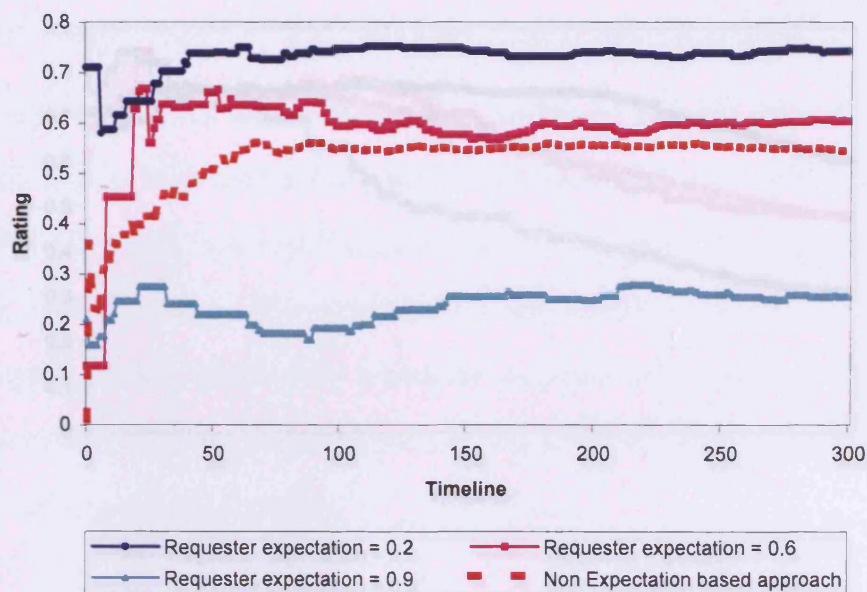
Figure 6.7: QoS Assessment for the Fluctuating Case

As can be seen, a fluctuating delivery of QoS had little impact on both expectation and non-expectation based assessment methods, except at the start of the experiment. This is because both methods average the individual ratings in assessment. If a different method is adopted for aggregating the individual ratings, for example, counting the number of ratings dropping below a specified expectation, then a different result can be expected. However, it remains a fact that our expectation based method is able to deliver the ratings in context, i.e., different requesters have different ratings for the same service, depending on their expectations. In Figure 6.8, we show the result of our test on the expectation based method using the delivered QoS data set with a downward trend.

It is interesting to observe in Figure 6.8 that our expectation based method reacts
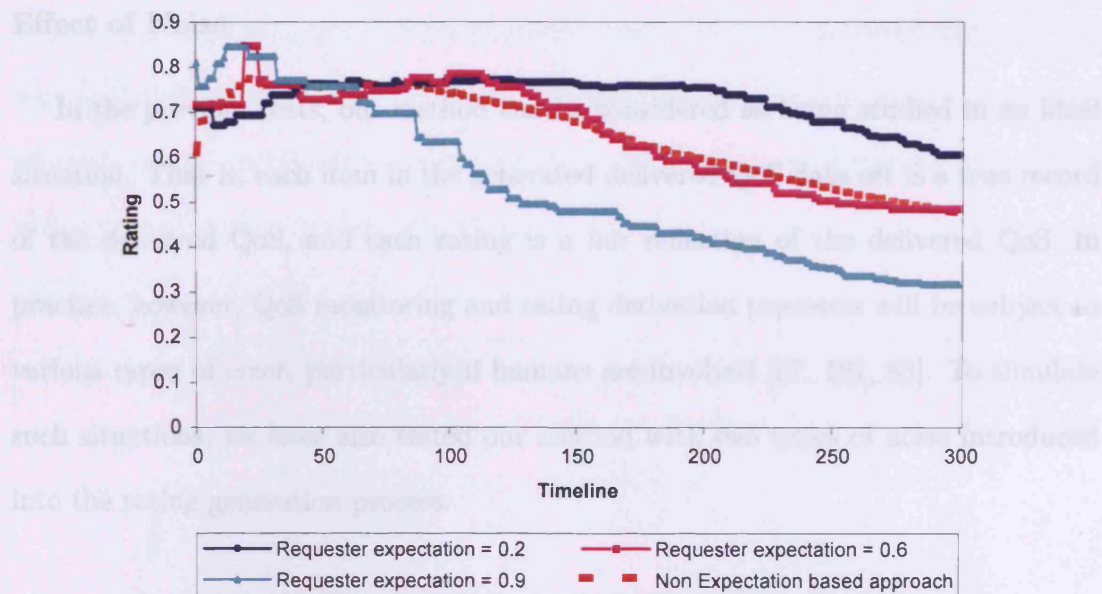
Figure 6.8: Sensitivity to QoS Changes

to the gradual drop of QoS differently, when the requester's expectations are different. For the requesters who have high expectations, this drop should affect their perception of the delivered QoS sooner. This is confirmed in Figure 6.8 which shows our method was able to pick up this fall much earlier than the non-expectation based one for this group of requesters. On the other hand, for requesters whose expectations are low, this gradual fall of QoS does not seem to have affected them as much. This again is understandable since as long as the QoS is above one's expectation, any fall from a very high QoS level to a lesser one should not really change one's view about the delivered QoS.

Effect of Noise

In the previous tests, our method can be considered as being studied in an ideal situation. That is, each item in the generated delivered QoS data set is a true record of the delivered QoS, and each rating is a fair reflection of the delivered QoS. In practice, however, QoS monitoring and rating derivation processes will be subject to various types of error, particularly if humans are involved [87, 181, 88]. To simulate such situations, we have also tested our method with two types of noise introduced into the rating generation process.
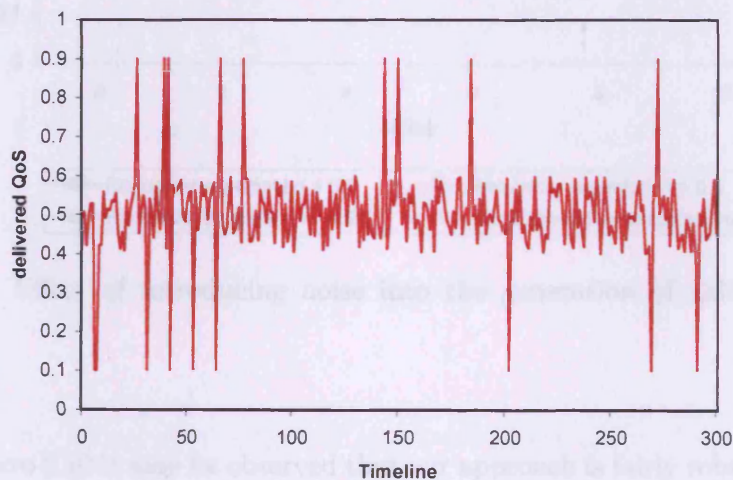

Figure 6.9: Near Constant Delivered QoS with Noise

In the first case, noise was introduced into the generation of delivered QoS. Such noise could be attributed to the errors in the QoS monitoring process. Figure 6.9 shows the near constant delivered QoS again, but with a small amount of noise in-

cluded. This kind of noise could obviously result in some distorted QoS ratings. Figure 6.10 shows the results of running both expectation and non-expectation based methods on the set of near constant delivered QoS data given in Figure 6.1, but with different percentages of noise introduced.
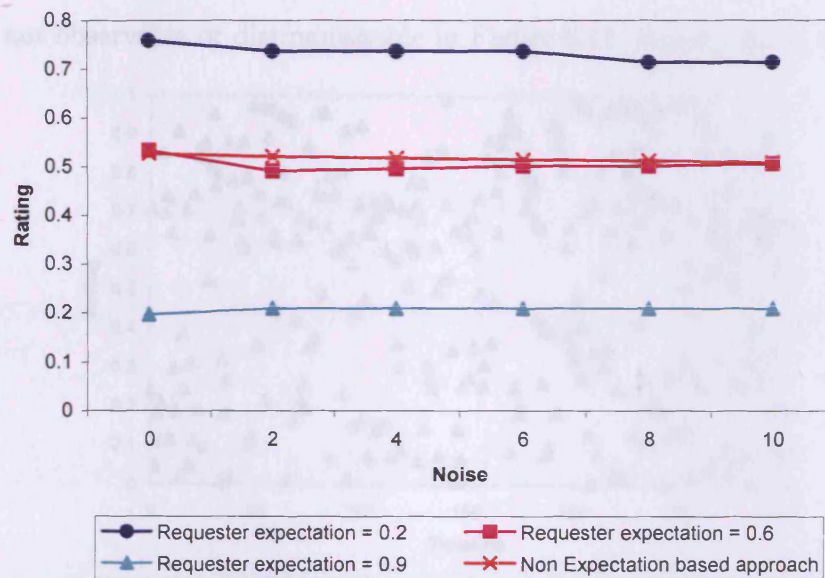
Figure 6.10: Effect of introducing noise into the generation of delivered QoS on assessment

From Figure 6.10 it may be observed that our approach is fairly robust in response to noise. The range of noise introduced in the experiments does not seem to deviate QoS assessment in any significant way, and our method still clearly delivers different ratings for requesters who have different expectations. We therefore believe that the method, we propose here can be used in real-world applications where totally accurate rating collection cannot be guaranteed.

In the second case, noise was introduced into the generation of QoS ratings. That

is, instead of following the rules given in Section 6.2.1 to generate all the ratings, we allow a percentage of the ratings to be generated with a degree of randomness. These randomly derived ratings can be regarded as representing some extremely harsh or lenient evaluations. Figure 6.11 below shows the ratings generated for the near constant delivered QoS, but with some noise introduced. Note that the introduced noise is not observable or distinguishable in Figure 6.11. Again, this is realistic.

Figure 6.11: Ratings for the Near Constant Delivered QoS with Noise

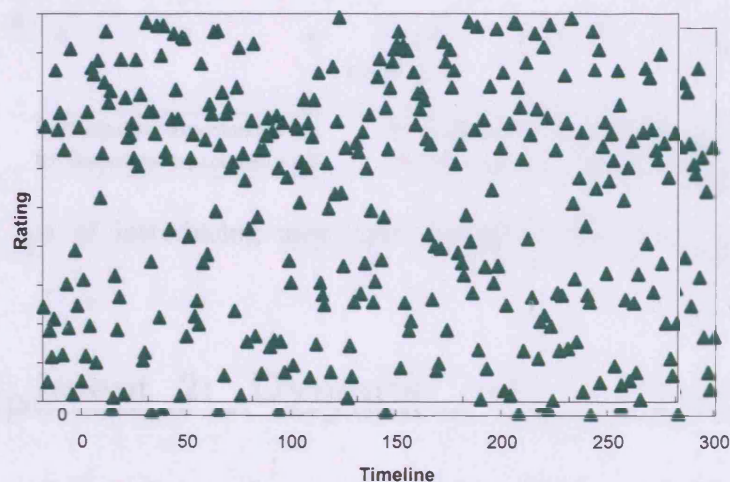We then ran both expectation and non-expectation based methods on a set of near constant delivered QoS data with different percentages of noise introduced. The results are shown in Figure 6.12. Again, the results confirm that the expectation based approach is robust in relation to noise.
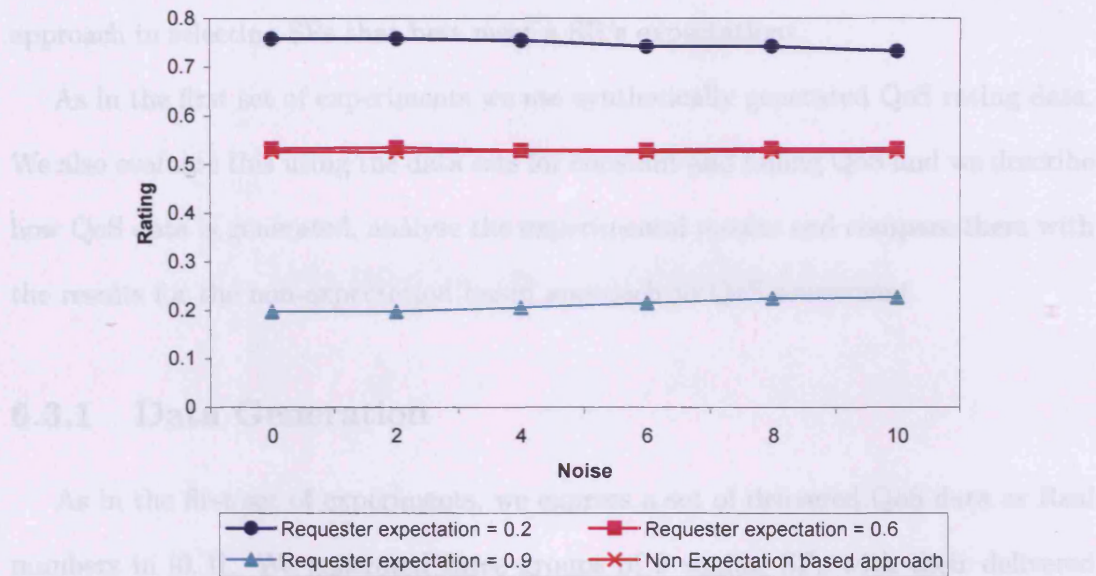
Figure 6.12: Effect of introducing noise into the generation of QoS ratings on assessment

6.3 Experiment 2: Dynamic selection of Service Provider based on Service Requesters Expectation

Ideally, the behaviour of SPs should be taken into account by QoS assessment algorithms. That is, the QoS assessment rating for an SP should change if the SP increases/decreases its QoS. Both the expectation and non-expectation based approaches should be able to detect changes in service behaviour, as both approaches use historical QoS rating data of a service for QoS assessment. However, we show that the expectation based approach is more effective in capturing and analysing the changes in service behaviour, by evaluating the accuracy of the expectation based

approach in selecting SPs that best meet a SR's expectations.

As in the first set of experiments we use synthetically generated QoS rating data. We also evaluate this using the data sets for constant and falling QoS and we describe how QoS data is generated, analyse the experimental results and compare them with the results for the non-expectation based approach to QoS assessment.

6.3.1 Data Generation

As in the first set of experiments, we express a set of delivered QoS data as Real numbers in $[0, 1]$. We generated three groups of 2 similar SPs with their delivered QoS displaying some specific patterns, which simulate some real world scenarios. For example, the sets of data shown in Figures 6.13, 6.14 and 6.15 exhibit three QoS levels: SP1 and SP2 at 0.1 - 0.3, SP3 and SP4 at 0.4 - 0.7 and SP5 and SP6 at 0.8 - 1.0, respectively, all indicating a fairly stable delivery of service.

In addition to the near constant delivered QoS data set, we have also tested our method on a case representing gradually falling QoS. This case is shown in Figure 6.16, where one of the SPs in each group in Figures 6.13, 6.14 and 6.15 has been set to have a gradual drop in QoS over a period of time. These two cases will allow us to study the effectiveness and robustness of our proposed expectation based QoS calculation method in some realistic settings.

We then generate three groups of expectations as before. For example, for the delivered QoS data given in Figures 6.13, 6.14 and 6.15, we generate three sets of expectations around $E(A) = 0.2$, $E(A) = 0.6$ and $E(A) = 0.9$, as shown in Figure 6.2. Finally, the ratings are generated for services, similar to as discussed in
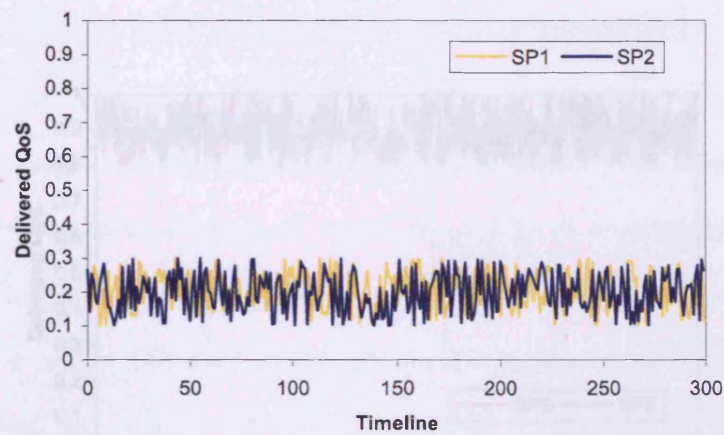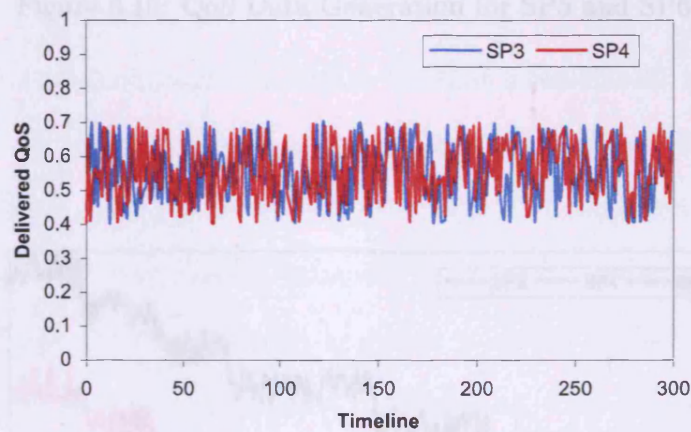
Figure 6.13: QoS Data Generation for SP1 and SP2

Figure 6.14: QoS Data Generation for SP3 and SP4

first experiment.

6.3.2 Results

The experiments are divided into two main blocks that correspond to two different scenarios. In the first scenario we consider SRs' expectations that can be met by an
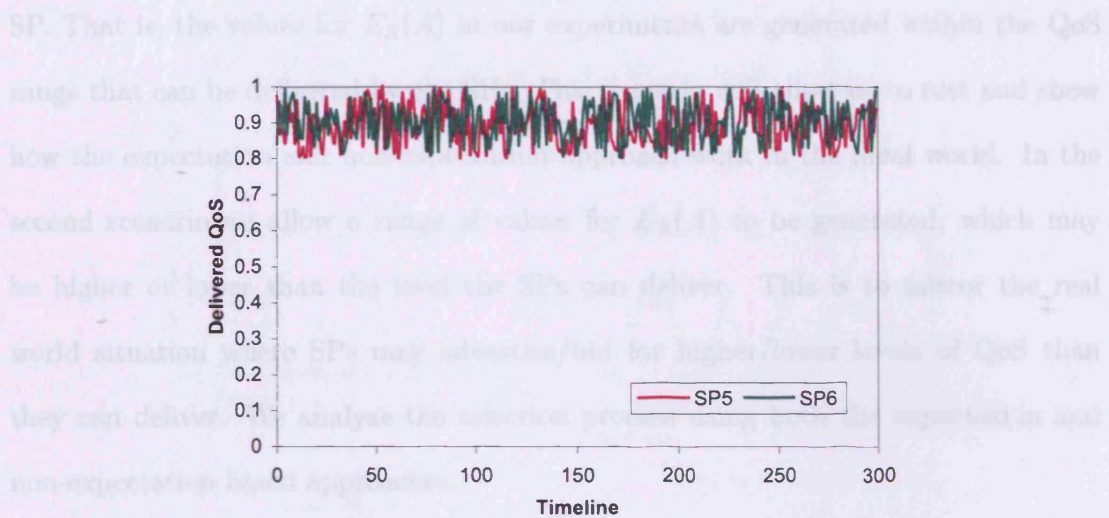
SP. That is, the values for $E_n(A)$ in our experiments are generated within the QoS range that can be delivered by the SPs. This allows the experiments to test and show how the expectation based approach behaves in a particular aspect of the real world. In the second scenario, we allow a range of values for $E_n(A)$ to be generated, which may be higher or lower than the level the SPs can deliver. This is to mirror the real world situation where SPs may advertise/bid for higher/lower levels of QoS than they can deliver. We analyse the selection process using both the expectation and non-expectation based approaches.

Figure 6.15: QoS Data Generation for SP5 and SP6

SCENARIO 1: SPs deliver near constant QoS

In the first set of experiments, we tried to establish a baseline for selecting services. We ran the service selection process using the near constant QoS delivery data sets (Figures 6.13, 6.14 and 6.15 for the entire duration of the experiments. The results shown in Figure 6.16 were derived using the expectation based approach and the performance of the non-expectation based approach is shown in Figure 6.18. In both approaches, one SP over the other would be selected. This is as expected because all SPs advertised a constant QoS and should have a relatively high QoS rating. Thus, in this case both the expectation and non-expectation approaches are equally good as they are able to select the service with the highest QoS delivery valuation.

Figure 6.16 and Figure 6.20 show the results of experiments under Figure 6.17 and Figure 6.18, but using the delivered QoS data set shown in Figure 6.16 with a downward trend.
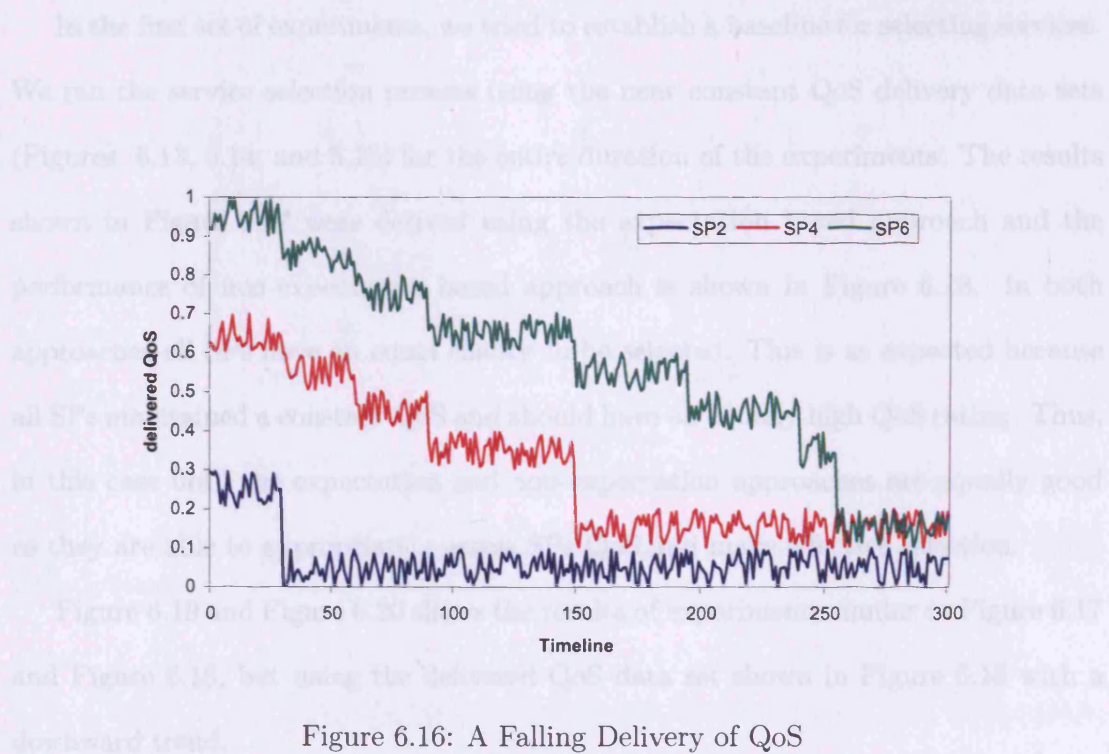

Figure 6.16: A Falling Delivery of QoS

In Figure 6.16 the experiment was conducted so that some SPs (SP2, SP4, and

SP. That is, the values for $E_R(A)$ in our experiments are generated within the QoS range that can be delivered by the SPs. This scenario will allow us to test and show how the expectation and non-expectation approach work in the ideal world. In the second scenario we allow a range of values for $E_R(A)$ to be generated, which may be higher or lower than the level the SPs can deliver. This is to mirror the real world situation where SPs may advertise/bid for higher/lower levels of QoS than they can deliver. We analyze the selection process using both the expectation and non-expectation based approaches.

SCENARIO 1 - SR's Expectations can be met by SPs

In the first set of experiments, we tried to establish a baseline for selecting services. We ran the service selection process using the near constant QoS delivery data sets (Figures 6.13, 6.14, and 6.15) for the entire duration of the experiments. The results shown in Figure 6.17 were derived using the expectation based approach and the performance of non-expectation based approach is shown in Figure 6.18. In both approaches all SPs have an equal chance to be selected. This is as expected because all SPs maintained a constant QoS and should have an equally high QoS rating. Thus, in this case both the expectation and non-expectation approaches are equally good as they are able to appropriately assess SPs QoS and make effective selection.

Figure 6.19 and Figure 6.20 shows the results of experiments similar to Figure 6.17 and Figure 6.18, but using the delivered QoS data set shown in Figure 6.16 with a downward trend.

In Figure 6.19 the experiment was conducted so that some SPs (SP2, SP4, and
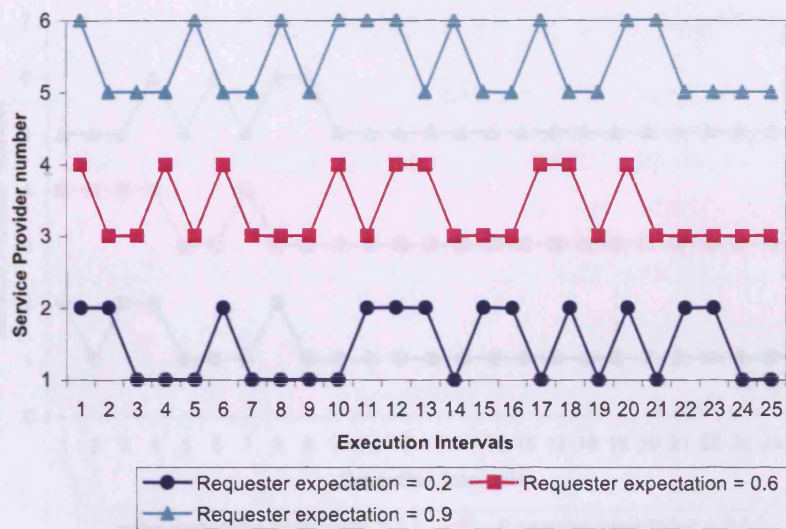
Figure 6.17: Expectation based approach: Service selection with near constant QoS delivery
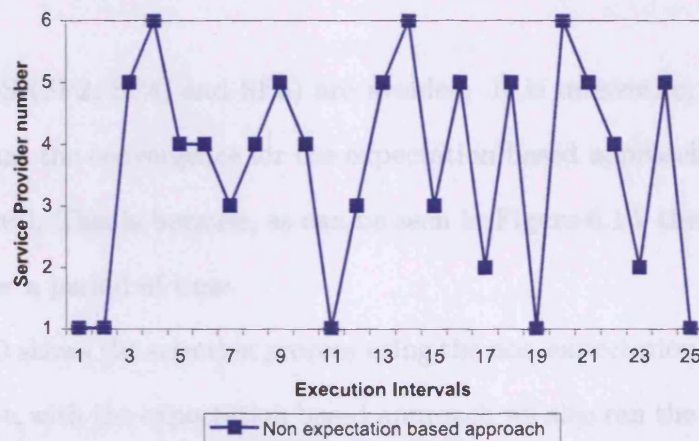

Figure 6.18: Non-expectation based approach: Service selection with near constant QoS delivery

SP6) gradually drop their delivered QoS. Figure 6.19 shows that for all three expectations ($E(A) = 0.2$, $E(A) = 0.6$ and $E(A) = 0.9$), the selection process converges to the SPs that maintain constant QoS (SP1, SP3, and SP5), while services that
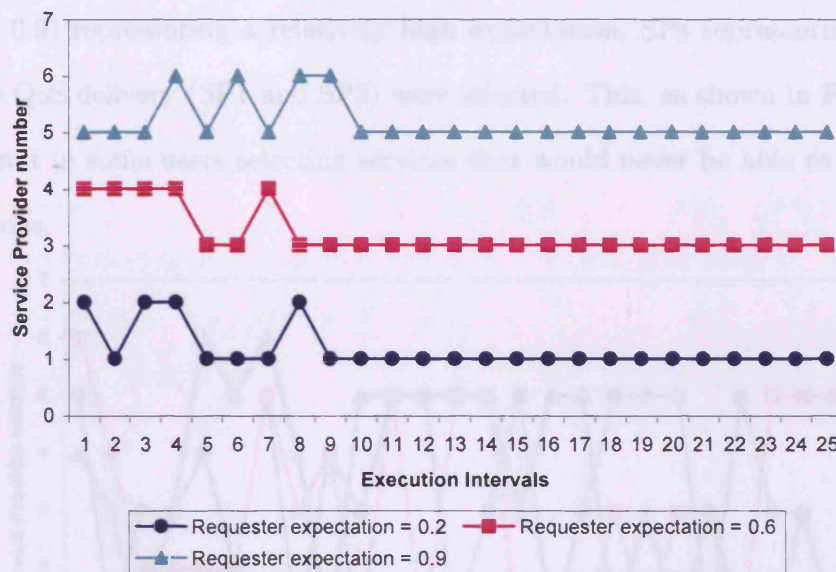
Figure 6.19: Expectation based approach: Service selection with falling QoS

drop their QoS (SP2, SP4, and SP6) are avoided. It is interesting to observe from Figure 6.19 that the convergence for the expectation based approach is delayed until the 10th interval. This is because, as can be seen in Figure 6.16, the fall in QoS only happened after a period of time.

Figure 6.20 shows the selection process using the non-expectation based approach. For comparison with the expectation based approach we also ran the non-expectation approach with three expectation groups, but the user expectation does not play any role in the selection process. Here, the selection process, in time (after 10th interval), finds the SPs that maintained a constant QoS (SP1, SP3, and SP5), while SPs that dropped their QoS (SP2, SP4, and SP6) were not selected. This is similar to the expectation based approach. However, if looked at closely each expectation and the service selected for that expectation, the results are quite different. For example, for

$(E(A) = 0.9)$ representing a relatively high expectation, SPs representing low and moderate QoS delivery (SP1 and SP3) were selected. This, as shown in Figure 6.20, would result in some users selecting services that would never be able to meet their expectations.
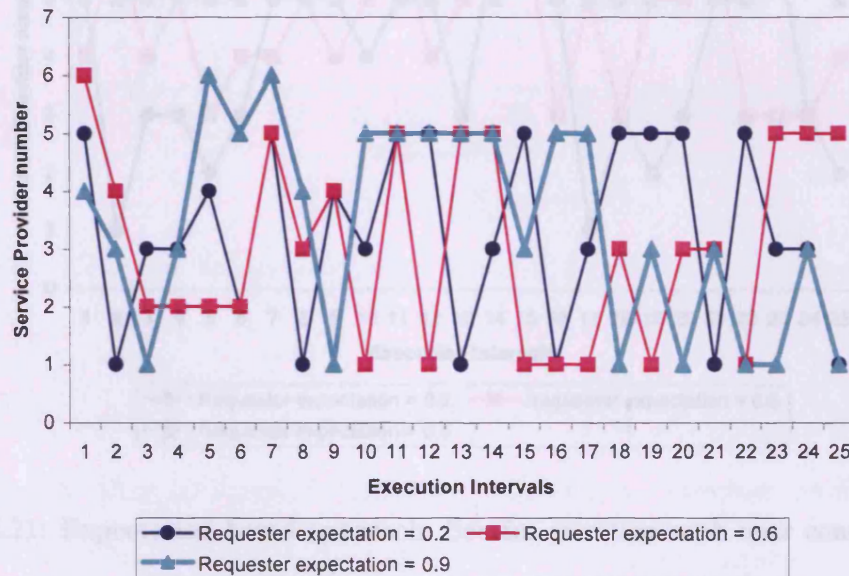

Figure 6.20: Non-expectation based approach: Service selection with falling QoS

Note also, that for users with low expectation, for example $E(A) = 0.2$, the non-expectation based approach, would not make a huge difference. This is understandable since as long as the QoS is above one's expectation, this should not change one's view about the delivered QoS.

SCENARIO 2 - SRs expectations may not be met by SPs

In the first set of experiments a range of SRs expectation values were generated that were higher/lower to the level that can be delivered by the SPs and were ex-

perimented with near constant QoS data sets. Figure 6.21 shows the results for this experiment.
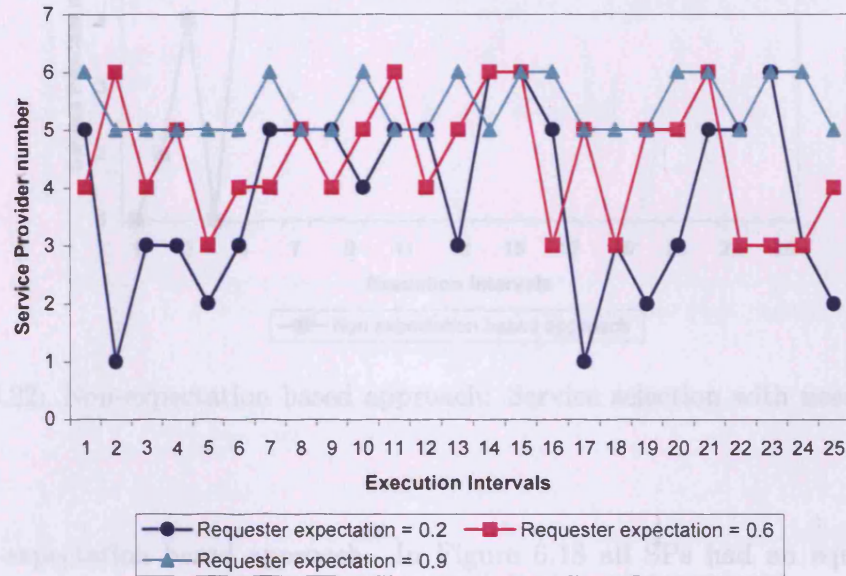
Figure 6.21: Expectation based approach: Service selection with near constant QoS

The results obtained in Figure 6.21 are very different to the ones shown in Figure 6.17 where user expectations were only generated within the SPs delivered QoS range. For requesters who have low expectations ($E(A) = 0.2$), the change allows them to now select SPs from all groups. Similarly, for the requesters whose expectations are moderate ($E(A) = 0.6$), the change allows them to select either from SP3 and SP4 or SP5 and SP6. On the other hand, for the requesters whose expectations are high ($E(A) = 0.9$), this change does not seem to have affected them as much. This reflects the real world scenario, where as long as the delivered QoS by an SP meets or exceeds one's expectation it will be considered an appropriate selection.

Figure 6.22 shows the experiment similar to that reported in Figure 6.21 but using
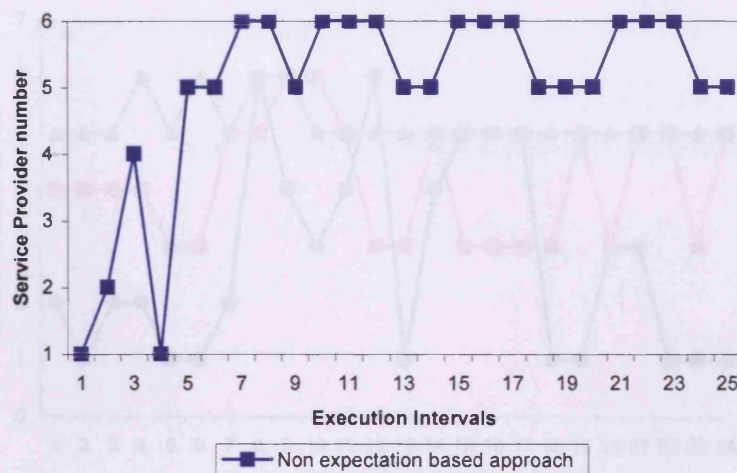
Figure 6.22: Non-expectation based approach: Service selection with near constant QoS

the non-expectation based approach. In Figure 6.18 all SPs had an equal chance to be selected, when SPs provide constant QoS delivery. However, in Figure 6.22 it can be seen that the case is quite different. The selection process is converged to only selecting from SP5 and SP6. This is due to the non-expectation approach penalizing SPs with low and moderate QoS delivery which do not meet certain SRs high expectations. Furthermore, since the non-expectation based approach averages all the past ratings, including ratings from expectations not required by the requester, the averages for SP1, SP2, SP3 and SP4 is always lower compared to that of SP5 and SP6. This approach to selection will disadvantage many SPs, even when they are capable of meeting SRs expectations.

Figures 6.23 and 6.24 show the results of experiments similar to those reported in Figures 6.21 and 6.22, but using the delivered QoS data sets with a downward trend. As can be seen in Figure 6.23, SPs that maintained a constant QoS (SP1, SP3 and
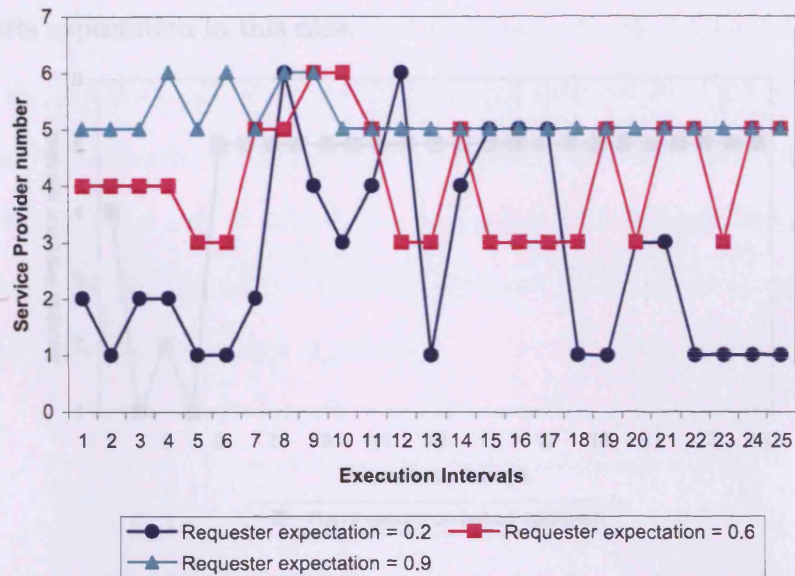
Figure 6.23: Expectation based approach: Service selection with falling QoS

SP5) got selected more often, while SPs that dropped their QoS levels (SP2,SP4 and SP6) were not so successful. Figure 6.23 shows that each expectation group converges to the better SP. It is interesting to observe from Figure 6.23 that our expectation based method reacts to the gradual drop of QoS differently when requesters' expectations are different. For the requesters who have low expectations ($E_R(A) = 0.2$), the QoS drop allows them to now select from all SPs. This again is understandable, since as long as the QoS is above one's expectation, any fall from a very high QoS level to a lesser one should not really change one's view about the delivered QoS. Similarly, for requesters whose expectations are moderate ($E_R(A) = 0.6$), the fall in QoS now allows them to select either from SP3, SP4, SP5 and SP6. On the other hand, for requesters whose expectations are high ($E_R(A) = 0.9$), this gradual fall does not seem to have affected them as much. This again is expected, as other SP groups could not

meet the SRs expectation in this case.
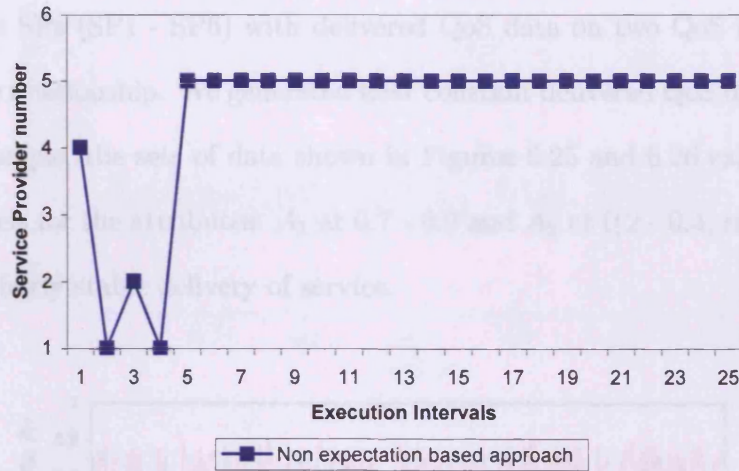
Figure 6.24: Non-expectation based approach: Service selection with falling QoS

Figure 6.24 shows the experiment which is similar to that reported in Figure 6.23 but uses the non-expectation based approach. In Figure 6.22 the falling QoS case resulted in convergence to SP5 and SP6. This, as we explained, was incorrect as it disadvantages many SPs, even when they are capable to meet users expectations. In Figure 6.24 similar results are shown, where the selection process converges to SP5. This shows that the non-expectation method is able to avoid SPs with falling QoS (SP2, SP4 and SP6) but also missed SPs that provide constant QoS (SP1 and SP3), which should have had an equal chance of being selected.

6.4 Fuzzy-based QoS assessment

In these experiments, we compare the Range-based QoS assessment algorithm to the Fuzzy-based QoS assessment introduced in Section 5.3.3. As in the first set of

experiments, we express a set of delivered QoS data as real numbers in $[0, 1]$. We generated six SPs (SP1 - SP6) with delivered QoS data on two QoS attributes displaying some relationship. We generated near constant delivered QoS data sets for all SPs. For example, the sets of data shown in Figures 6.25 and 6.26 exhibit two QoS delivered levels for the attributes: A_1 at 0.7 - 0.9 and A_2 at 0.2 - 0.4, respectively, all indicating a fairly stable delivery of service.
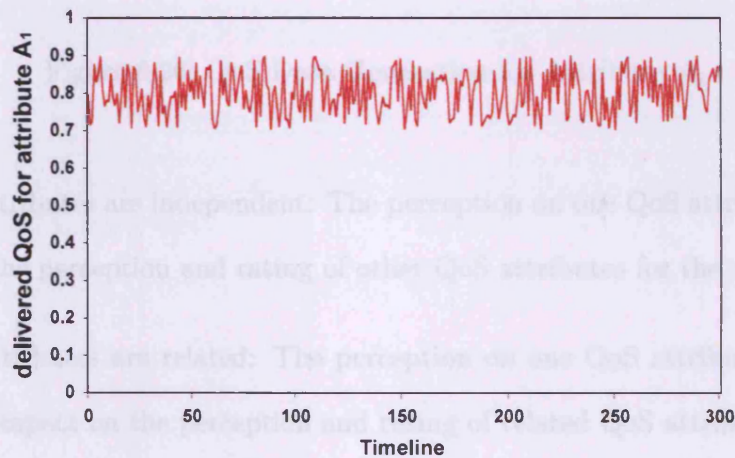
Figure 6.25: QoS Data Generation for Attribute A_1

Similar to the experiment shown in Figure 6.21, this experiment introduces a range of SRs expectations for each SP as part of the simulated QoS data. All QoS assessment experiments were performed with SRs expectations $E(A_1) = 0.8$ and $E(A_2) = 0.3$.

We considered the following two QoS attribute relationship scenarios in the experiments:
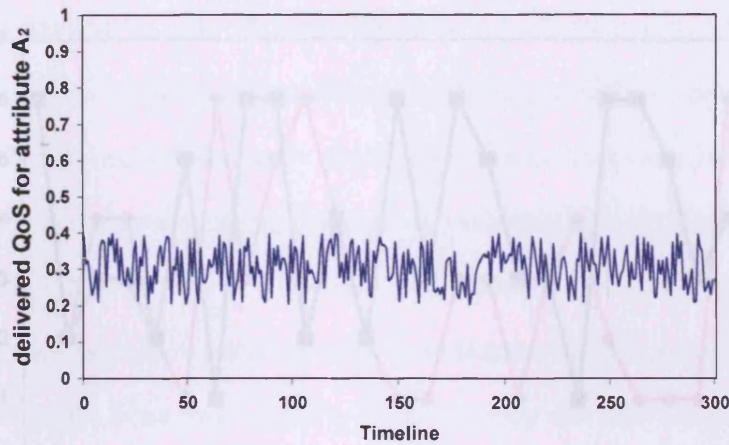
Figure 6.26: QoS Data Generation for Attribute A_2

1. QoS attributes are independent: The perception on one QoS attribute does not affect the perception and rating of other QoS attributes for the service; and

2. QoS attributes are related: The perception on one QoS attribute will have a direct impact on the perception and rating of related QoS attributes.

Figure 6.27 shows the result of the experiment using the first rating scenario, and Figure 6.28 the second. To simulate the effect of relationship in Figure 6.28 the experiment was conducted such that for some SPs (SP2, SP4, and SP6) lower than expected perception on A_1 caused low QoS ratings on A_2.

The experiments were conducted in such a way that all SPs can meet requested expectations (i.e., $E(A_1) = 0.8$ and $E(A_2) = 0.3$), thus all SPs should stand an equal chance of being selected. In Figure 6.27 it can be seen that the results are as expected, i.e., all SPs have been selected. This result suggests that both algorithms produce a similar service selection pattern and thus are equally effective. However,
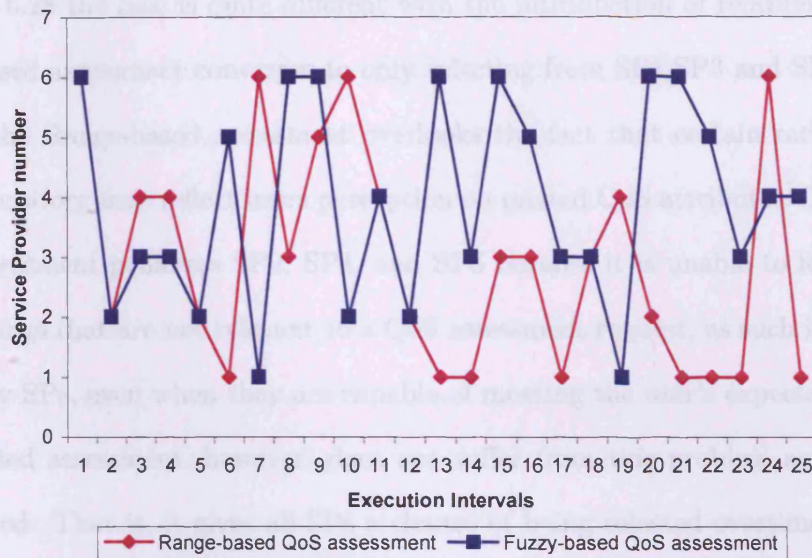
Figure 6.27: Service selection with Fuzzy-based and Range-based QoS assessment using independent attributes
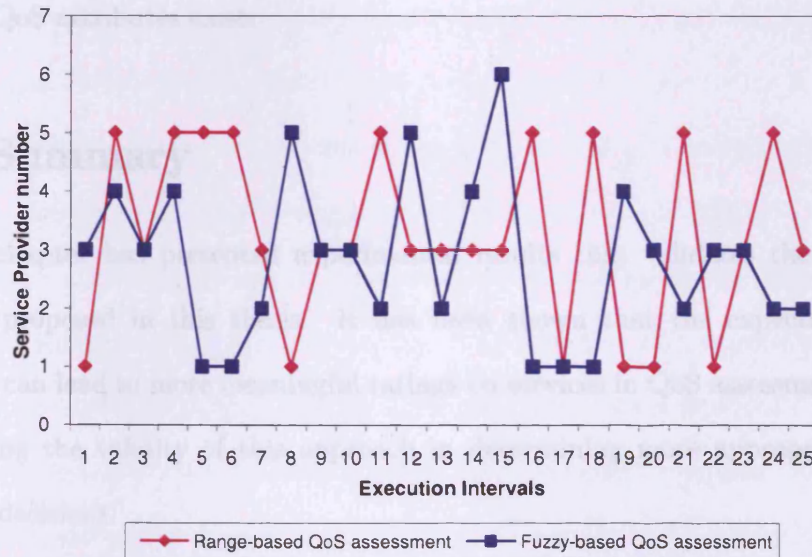
Figure 6.28: Service selection with Fuzzy-based and Range-based QoS assessment using related attributes

in Figure 6.28 the case is quite different with the introduction of relationships. The Range-based assessment converges to only selecting from SP1,SP3 and SP5. This is because the Range-based assessment overlooks the fact that certain ratings in the rating repository may reflect users perception on related QoS attributes. The Range-based assessment penalizes SP2, SP4, and SP6 because it is unable to identify and reject ratings that are not relevant to a QoS assessment request, as such it disadvantage many SPs, even when they are capable of meeting the user's expectations. The Fuzzy-based assessment, however, does not suffer from this problem and performs as expected. That is, it gives all SPs a chance of being selected overtime. So, both the Fuzzy-based and Range-based QoS assessment are equally viable approaches to service selection where services' QoS attributes are independent, but, Fuzzy-based assessment proves to be a better candidate for service selection when relationships between QoS attributes exist.

6.5 Summary

This chapter has presented experimental results that validates the theoretical concepts proposed in this thesis. It has been shown that the expectation based approach can lead to more meaningful ratings on services in QoS assessment, thereby establishing the validity of this approach in determining more appropriate service selection decisions.

We have experimentally demonstrated that our expectation based approach allows QoS ratings to be dynamically generated by matching expectations from historical QoS data with that of SRs expectations. As a result, a personalized QoS assessment

is established for the SR. This is in contrast to the non-expectation based approach which returns a fixed QoS verdict, implying that different SRs would all agree with this fixed assessment. By producing ratings for SRs on a case by case basis, our approach produce ratings that are meaningful to and will be agreed by the individual SRs.

We have also experimentally demonstrated that our service selection process is able to accurately identify and adapt to changes in delivered QoS. Using our approach, services that provide constant QoS get selected often, while services, whose delivery of QoS deteriorates, are rejected often in the selection process. In contrast, the results of the non-expectation based approach shows that it disadvantages many services from selection, even when they are capable of meeting SRs expectations.

We also experimentally compared our Range-based and Fuzzy-based QoS assessment for service selection using multiple attributes, where; (i) the attributes are independent, that is they are not related to each other, and (ii) the attributes are related. The results demonstrate that, in case of the independent attributes both algorithms produce a similar service selection pattern and thus are equally effective. However, in presence of relationship among the attributes, the Range-based assessment disadvantages many services even when they are capable of meeting SRs expectations. This is because it is unable to identify related attributes, and thus is unable to select/reject ratings appropriately that are relevant to a QoS assessment request. Whereas, the Fuzzy-based assessment results in selection of all services that meets SRs expectations, and thus, proves to be a better candidate for service selection in presence of related attributes.

Overall, the empirical evaluation highlights the ability of our expectation based approach to perform better than non-expectation approach, since it enabled consumers to identify which services are most reliable, and which are best avoided based on their past performances. Thus, we successfully met the aims in the hypothesis specified in Section 1.1.

Chapter 7

Conclusion and Future Work

7.1 Research Summary

The SOC paradigm has offered a great potential for creating a marketplace where autonomous trade can be facilitated between service providers and service consumers using a dynamic service discovery and selection process. The common technique to facilitate service discovery is based on functional matching of service requests with service advertisements. Current approaches do not consider service selection after service discovery, and it is assumed that the service consumer will be satisfied with any of the discovered services. Such a mechanism has some serious limitations if the true potential of an SOC-powered e-commerce environment is to be realized. It is now well accepted in SOC research that non-functional aspects such as QoS are equally important to consider in selection. Although studying QoS in computing research is not new, research in QoS so far has mostly considered this aspect, from the low level resource optimization point of view, for example network bandwidth and CPU

allocation. In an open marketplace, QoS also needs to be studied at the application level. This will help SRs discover and select SPs that will best meet their QoS requirements. There are three major issues concerning incorporating of QoS into the SOC paradigm. First, QoS needs to be represented using flexible and extensible schemas that allow SPs to describe their QoS offerings along with a technical description that allows SRs to specify their QoS requirements. Second, a mechanism to register and discover services with QoS constraints is needed. Third, there is a need to establish a mechanism for QoS assessment that can accurately rank services for user. This thesis has addressed these three issues with positive results.

7.2 Research Contributions

The thesis addresses key issues in incorporating QoS into a service-based environment, namely, QoS-aware specification of service advertisements and requirements, discovery of services based on specified QoS requirements, and QoS assessment of services to support selection among competing services. This thesis has contributed to the state-of-the-art of QoS support in service discovery and selection in a broad and non domain-specific way. This section describes its main research contributions.

- *Development of a QoS incorporated architectural model for SOC*: The thesis has proposed a QoS model that supports the requirements of a QoS-aware e-commerce facility in an SOC environment. These requirements ensure QoS-awareness during the whole e-commerce life-cycle, namely representing QoS, discovering suitable services, and dynamically assessing QoS based on a SRs expectations. We demonstrated the applicability of our model using a VO

formation application which mirrors service discovery and selection in the real world.

- *Development of QoS incorporated service description for service discovery*: Although there exist many approaches to service description, such as WSDL and UDDI, they do not provide support for incorporating QoS information in the description. There is no standard for representing QoS information, especially when expressing service requirements and advertisements for the purpose of service discovery. Some approaches exist that represent QoS in service description, but these techniques are either domain-specific or not relevant to service discovery. This thesis proposed a quality schema, which is at the core of our service discovery framework. All components involved in the service discovery process such as requesters, providers, discovery agent, and advertisement repository apply and utilizes this quality schema. The proposed schema is simple to use, fully extensible, and standard conforming. These properties of the schema are due to the use of DAML-S semantics, which enable the addition of new QoS parameters with little or no change in the service discovery component (i.e., the YP agent). In addition, the schema also supports representation of a wide variety of QoS attributes, from service independent attributes such as response time and availability to domain specific attributes such as frame rate for a multimedia domain. The schema provides a framework which allows addition of any domain specific or customised properties in a format that is simple and interpretable by humans.

- *Development of the YP agent to support QoS-aware service discovery*: Often in a service-based environment, many services exist that provide the required functionality but having different levels of QoS. Moreover, not all QoS parameters are of the same importance to a service requester. This necessitates a mechanism to incorporate QoS into service discovery process that takes into account the desired and offered levels of QoS. This thesis proposed a YP agent that is capable of discovering the most suitable service advertisements based on the SRs functional and QoS requirements. The advertisements and requirements are described using the QoS schema and is processed by the YP agent for autonomous service discovery.

- *Development of the QoS agent for QoS assessment*: Most other QoS models incorporating service selection studies have been developed in the context of optimizing certain parameters, such as completion time, for a chain of services. Some approaches exist based on calculating trust and reputation to assist consumers in service selection. However, these systems are constructed without a formal QoS model. Most of these systems are based on building a rating system which calculates trust or reputation of a SP without much regard to the context or reasons behind the collected user ratings. This thesis proposed an approach to performing QoS assessment that is firmly rooted in the popular and well accepted findings from business studies about using the context behind QoS rating. In particular, our approach makes explicit the importance of users expectations in making meaningful QoS assessment. Based on the expectation-based approach, we have presented two QoS assessment algorithms, that formulate

a personalized QoS assessment for an SR. The personalized QoS assessment is made based on selecting only the ratings that have similar expectations to that which the service requester expects. We have demonstrated the effectiveness of our QoS assessment approach in service selection using a simulated dataset. Our experiments show that the proposed approach can lead to more meaningful ratings on services - enabling SRs to make more appropriate service selection decisions.

7.3 Research Directions

The thesis has focussed on incorporating QoS into service discovery and selection in an SOC environment. There are several more extensions possible to the QoS based service discovery framework. One of them is to perform a Web service orchestration, i.e., to find a chain of different services necessary to execute a Web service workflow. In this thesis we modelled a single YP agent to which all SRs communicate the service discovery requests. It will be an interesting challenge to consider the case where multiple YP components, either distributed or working as a cluster, are used, and study how they may collaborate, especially if we allow each YP component to have its own local domain service and quality ontologies.

Motivated by the idea of expectation, we derive a QoS rating dynamically at the time of assessment, using only the ratings that meet a certain context. This approach to searching for users that share similar likes and dislikes, is closely related to the increasingly active area of recommendations in social networking (e.g., Movie-Lens [67], iLike [80]). We intend to study different aspects of similarity assessment

based on social networking models, to develop an integrated approach to providing a way to handle QoS assessment failures, i.e., when no QoS information with similar expectations is available.

We considered the collection of QoS ratings from users based on the assumption that the information provided is reliable and trustworthy. When untrustworthy/malicious users are present in the system, problems may arise for the rating calculator. Enhancing the rating collector to verify ratings by detecting deceitful users, is another area of research to be undertaken.

In our implementation of the rating calculator we experimented with several algorithms for QoS assessment that take the reasons behind a user's rating into account. It would be interesting to investigate if it is possible to share such contextual QoS information among other rating assessment agencies (e.g., eBay, Amazon, and epinions) to exploit a wide variety of mechanisms/algorithms for QoS rating calculation. Based on this we could study, if a combination of QoS ratings produced from different QoS assessment algorithms can be better than our current method.

The QoS perspective adopted in this thesis is a rather conformance-centric one, and is useful mainly from the QoS assurance point of view. That is, we implicitly assume that if we choose a service that has a maximum conformance quality for a consumer, then the service will be appreciated by the consumer, regardless of who the consumer is. In practice, however, this may not be the case. Quality is highly related to consumer satisfaction. For example, while the majority of people may not mind missing one news update between 2-5am due to some system failure, hence still consider that the service has good conformance quality in terms of reliability, taxi

drivers who are on night shifts might view it differently. So, for a QoS assessment system to be truly useful, the challenge is to investigate a consumer-centric QoS model. The expectation based QoS approach we suggested here is general enough to support the exploration of such a model.

Reference

[1] EbXML. http://www.ebxml.org/.

[2] OWL-S. http://www.daml.org/services/owl-s/1.0/.

[3] UDDI Version 2 API Specification. http://uddi.org/pubs/ProgrammersAPI-V2.04-Published-20020719.htm.

[4] Web Services Description Language (WSDL). http://www.w3.org/TR/wsdl.

[5] Alfarez Abdul-Rahman and Stephen Hailes. A distributed trust model. In *NSPW '97: Proceedings of the 1997 workshop on New security paradigms*, pages 48–60, New York, NY, USA, 1997. ACM Press.

[6] R. J. Al-Ali, O. F. Rana, D. W. Walker, S. Jha, and S. Sohail. G-QoSM: Grid service discovery using QoS properties. *Journal of Computing and Informatics (Special issue on Grid Computing)*, 21(4):363–382, 2002.

[7] Kamal Ali and Wijnand Van Stam. Tivo: making show recommendations using a distributed collaborative filtering architecture. In *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 394–401, New York, NY, USA, 2004. ACM Press.

141

[8] Amazon. http://www.amazon.com.

[9] A. Ankolekar, M. Burstein, J. Hobbs, O. Lassila, D. Martin, S. McIllraith, S. Narayanan, M. Paolucci, T. Payne, K. Sycara, and H. Zeng. DAML-S: Web service description for the Semantic Web. In *Proceedings of International Semantic Web Conference (ISWC)*, 2002.

[10] K. Arabshian and H. Schulzrinne. GloServ: Global service discovery architecture. In *The First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services*, 2004.

[11] Mark Arunasalam, Albert Paulson, and William Wallace. Service quality assessment of workers' compensation health care delivery programs in new york state using servqual. *Health Marketing Quarterly*, 21(1-2), 2003.

[12] C. Aurrecoechea, A. T Campbell, and L. Hauw. A survey of qos architectures. *Multimedia Systems Journal*, 6(3):138–151, 1998.

[13] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.

[14] Nisreen N. Bahnan. *A dimensional approach to understanding the variation in consumers' ratings of service quality.* PhD thesis, Temple University, 2003.

[15] Ranieri Baraglia and Fabrizio Silvestri. An online recommender system for large web sites. In *Web Intelligence*, pages 199–205, Los Alamitos, CA, USA, 2004. IEEE Computer Society.

[16] Luciano Baresi, Sam Guinea, and Pierluigi Plebani. WS-Policy for service monitoring. In *Technologies for E-Services*, 2006.

[17] Allen Lawrence Barker. *Selection of distance metrics and feature subsets for kNN classifiers*. PhD thesis, Department of Computer Science, University of Virginia, 1997.

[18] Chumki Basu, Haym Hirsh, and William W. Cohen. Recommendation as classification: Using social and content-based information in recommendation. In *AAAI/IAAI*, pages 714–720, 1998.

[19] B. Benatallah, Q. Z Sheng, and M. Dumas. The Self-Serv environment for Web services composition. *Internet Computing*, 7(1):40 – 48, Jan-Feb 2003.

[20] Leonard L Berry and A Parasuraman. Listening to the customer – the concept of a service-quality information system. *Sloan Management Review*, 38(3), 1997.

[21] A. S Bilgin and M. P Singh. A DAML-based repository for QoS-aware semantic Web service selection. In *Proceedings of the IEEE International Conference on Web Services*, 2004.

[22] Yolanda Blanco-Fernandez, Jose J. Pazos-Arias, Alberto Gil-Solla, Manuel Ramos-Cabrer, Belen Barragans-Martinez, and Martin Lopez-Nores. A multi-agent open architecture for a tv recommender system: A case study using a bayesian strategy. In *ISMSE '04: Proceedings of the IEEE Sixth International Symposium on Multimedia Software Engineering (ISMSE'04)*, pages 178–185, Washington, DC, USA, 2004. IEEE Computer Society.

[23] Laura Bocchi, Paolo Ciancarini, Rocco Moretti, Valentina Presutti, and Davide Rossi. An OWL-S based approach to express grid services coordination. In *SAC '05: Proceedings of the 2005 ACM symposium on Applied computing*, pages 1661–1667, New York, NY, USA, 2005. ACM Press.

[24] D. Booth, H. Haas, F. McCabe, E. Newcomer, M. Champion, C. Ferris, and D. Orchard. Web services architecture. http://www.w3.org/TR/ws-arch/, Feburary 2004.

[25] Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, and Franois Yergeau. Extensible markup language (XML) 1.0. http://www.w3.org/TR/xml, September 2006.

[26] S. Braynov and T. Sandholm. Incentive compatible mechanism for trust revelation. In *Proceedings of First International Conference on Autonomous Agents and Multi Agent Systems*, pages 310–311, 2002.

[27] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *In Proceedings of the Fourteenth Annual Conference on Uncertainty in Artificial Intelligence*, pages 43–52, 1998.

[28] Paul A. Buhler and Jose M. Vidal. Towards adaptive workflow enactment using multiagent systems. *Inf. Tech. and Management*, 6(1):61–87, 2005.

[29] M. Burgess, W. A Gray, and N. Fiddian. Establishing a taxonomy of quality for use in information filtering. In *Proceedings of 19th British National Conference on Databases*, pages 103–113, 2002.

[30] Robin Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, 2002.

[31] Gerardo Canfora, Massimiliano Di Penta, Raffaele Esposito, and Maria Luisa Villani. An approach for QoS-aware service composition based on genetic algorithms. In *GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation*, pages 1069–1075, New York, NY, USA, 2005. ACM Press.

[32] Anthony J Capraro and Robert D Yearout. Selecting quality initiatives to pursue: Integrating demand effects into the evaluation and selection of potential quality initiatives. *Journal of Marketing Theory and Practice*, 12(3), 2004.

[33] J. Cardoso, A. Sheth, J. Miller, J. Arnold, and K. Kochut. Quality of service for workflows and Web service processes. *Web Semantics*, 1(3):281–308, April 2004.

[34] F. Casati, S. Ilnicki, J. Li-Jie, V. Krishnamoorthy, and S. Ming-Chien. An open, flexible, and configurable system for service composition. In *Second International Workshop on Advanced Issues of E-Commerce and Web-Based Information Systems (WECWIS 2000)*, pages 125 – 132, April 2000.

[35] F. Casati and M. C Shan. Definition, execution, analysis, and optimization of composite e-services. *Data Engineering Bulletin*, 4(1):29–34, 2001.

[36] D. Chalmers and M. Sloman. A survey of quality of service in mobile computing environments. *IEEE Communications Surveys and Tutorials*, 2(2):2–10, 1999.

[37] Anthony Chavez and Pattie Maes. Kasbah: An agent marketplace for buying and selling goods. In *Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM'96)*, 1996.

[38] Z. Chen, Chia Liang-Tien, B. Silverajan, and Lee Bu-Sung. UX: an architecture providing QoS-aware and federated support for UDDI. In *Proceedings of the International Conference on Web Services*, 2003.

[39] Christina Christakou and Andreas Stafylopatis. A hybrid movie recommender system based on neural networks. In *Proceedings of the 5th International Conference on Intelligent Systems Design and Applications (ISDA)*, pages 500–505, Los Alamitos, CA, USA, 2005. IEEE Computer Society.

[40] Erik Christensen, Francisco Curbera, Greg Meredith, and Sanjiva Weerawarana. Web services description language (WSDL) 1.1. http://www.w3.org/TR/wsdl, March 2001.

[41] Workflow Management Coalition. The workflow reference model. http://www.wfmc.org/standards/docs/tc003v11.pdf, Jan 1995.

[42] Carlo Combi and Giuseppe Pozzi. Architectures for a temporal workflow management system. In *Proceedings of the 2004 ACM symposium on Applied computing*, pages 659–666. ACM Press, 2004.

[43] OASIS Technical Committee. Universal description, discovery and integration (UDDI). http://www.uddi.org/specification.html, August 2005.

[44] Asit Dan, Catalin Dumitrescu, and Matei Ripeanu. Connecting client objectives with resource capabilities: an essential component for grid service managent infrastructures. In *Proceedings of the 2nd international conference on Service oriented computing(ICSOC'04)*, pages 57–64, New York, NY, USA, 2004. ACM Press.

[45] Harold Davis. *Google Advertising Tools: Cashing in with AdSense, AdWords, and the Google APIs*. O'Reilly Media, Inc., 2006.

[46] Chrysanthos Dellarocas. Analyzing the economic efficiency of ebay-like online reputation reporting mechanisms. In *EC '01: Proceedings of the 3rd ACM conference on Electronic Commerce*, pages 171–179, New York, NY, USA, 2001. ACM Press.

[47] ShuiGuang Deng, Zhen Yu, ZhaoHui Wu, and LiCan Huang. Enhancement of workflow flexibility by composing activities at run-time. In *Proceedings of the 2004 ACM symposium on Applied computing(SAC'04)*, pages 667–673, New York, NY, USA, 2004. ACM Press.

[48] V. Deora, J. Shao, W. A Gray, and N. J Fiddian. A quality of service management framework based on user expectations. In *Proceedings of the First International Conference on Service Oriented Computing (ICSOC)*, pages 104–114, Italy, December 2003.

[49] V. Deora, J. Shao, W. A Gray, and N. J Fiddian. Expectation based quality of service assessment. *Journal on Digital Libraries*, 6(3):260–269, 2006.

[50] A. Dogac, I. Cingil, G. B Laleci, and Y. Kabak. Improving the functionality of uddi registries through web service semantics. In *3rd VLDB Workshop on Technologies for E-Services (TES-02)*, pages 9–18, Hong Kong, China, August 2002.

[51] A. Dogac, Y. Kabak, and G. Laleci. Enriching ebXML registries with OWL ontologies for efficient service discovery. In *14th International Workshop on Research Issues on Data Engineering*, Boston, USA, March 2004.

[52] Prashant Doshi, Richard Goodwin, Rama Akkiraju, and Kunal Verma. Dynamic workflow composition using markov decision processes. In *Proceedings of the IEEE International Conference on Web Services (ICWS'04)*, pages 576–582, Washington, DC, USA, 2004. IEEE Computer Society.

[53] Ebay. http://www.ebay.com.

[54] Larry P. English. *Improving Data Warehouse and Business Information Quality*. Wiley & Sons, 1999.

[55] Siddharth Bajaj et al. Web services policy framework (WS-Policy). http://www-128.ibm.com/developerworks/library/specification/ws-polfram/, March 2006.

[56] S. Frlund and J. Koistinen. Quality-of-Service specification in distributed object systems, 1998.

[57] S. Frolund and J.Koistinen. QML: A language for quality of service specification. Technical report, HP Laboratories, 1998.

[58] Michael R. Genesereth and Steven P. Ketchpel. Software agents. *Communications of the ACM*, 37(7), 1994.

[59] Thomas George and Srujana Merugu. A scalable collaborative filtering framework based on co-clustering. In *Proceedings of the Fifth IEEE International Conference on Data Mining(ICDM'05)*, pages 625–628, Washington, DC, USA, 2005. IEEE Computer Society.

[60] Michael Gillmann, Gerhard Weikum, and Wolfgang Wonner. Workflow management with service quality guarantees. In *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, pages 228–239. ACM Press, 2002.

[61] David Goldberg, David Nichols, Brian M. Oki, and Douglas Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, 1992.

[62] N. Good, T. Schafer, J. Konstan, A. Borchers, B. Sarwar, J.Herlocker, and J. Riedl. Combining collaborative filtering with personal agents for better recommendations. In *In Proceedings of the 1999 Conference of the American Association of Artifical Intelligence (AAAI-99)*, 1999.

[63] Google. www.google.com.

[64] Jerry B Gotlieb and Dawn Langkamp Bolton. Three of the key variables that help explain how consumers develop their perception of the quality of organi-

zations marketing services. *Journal of Professional Services Marketing*, 21(1), 2000.

[65] D. Greising. Quality: How to make it pay. *Business Week*, 1994.

[66] Grid Resources for Industrial Application Project. Configuring the SLA service. http://archive.gria.org/docs/5.0.1/user-guide/service-provider-management/sla/config.html, 2005.

[67] MovieLens Group. http://movielens.umn.edu/login.

[68] W3C XML Protocol Working Group. SOAP Version 1.2. http://www.w3.org/TR/soap/, 2006.

[69] X. Gu. An XML-Based quality of service enabling language for the web. *Journal on Visual Language and Computing*, 3(1):61 - 95, Feb 2000.

[70] R. Guha, Ravi Kumar, Prabhakar Raghavan, and Andrew Tomkins. Propagation of trust and distrust. In *Proceedings of the 13th international conference on World Wide Web (WWW'04)*, pages 403–412, New York, NY, USA, 2004. ACM Press.

[71] Robert J. Hall and Andrea Zisman. Behavioral models as service descriptions. In *Proceedings of the 2nd international conference on Service oriented computing (ICSOC04)*, pages 163–172, New York, NY, USA, 2004. ACM Press.

[72] R. Hallowell. The relationships of customer satisfaction, customer localty and profitability: an empirical study. *International Journal of Service Industry Management*, 1996.

[73] Jun He, Matti A. Hiltunen, Mohan Rajagopalan, and Richard D. Schlichting. Providing QoS customization in distributed object systems. In *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg (Middleware'01)*, pages 351–372, London, UK, 2001. Springer-Verlag.

[74] M. He, N. R Jennings, and H. Leung. On agent-mediated electronic commerce. *IEEE Trans on Knowledge and Data Engineering*, 15(4):985–1003, 2003.

[75] J. Herlocker, J. Konstan, and J. Riedl. Explaining collaborative filtering recommendations. In *Proceedings of ACM 2000 Conference on Computer Supported Cooperative Work*, pages 241–250, 2000.

[76] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22(1):5–53, 2004.

[77] Y. Hoffner, H. Ludwig, P. Grefen, and K. Aberer. CrossFlow: integrating workflow management and electronic commerce. *SIGecom Exchange*, 2(1):1–10, 2001.

[78] Wolfgang Hoschek. The web service discovery architecture. In *Proceedings of the 2002 ACM/IEEE conference on Supercomputing(Supercomputing'02)*, pages 1–15, Los Alamitos, CA, USA, 2002. IEEE Computer Society Press.

[79] Edward U Bond III and Ross L Fink. Customer satisfaction and the marketing-quality interface. *The Journal of Business & Industrial Marketing. Santa Barbara*, 18(2-3), 2003.

[80] iLike Team. http://www.ilike.com/.

[81] Naceur Jabnoun, Hussein A Hassan, and Al-Tamim. Measuring perceived service quality at uae commercial banks. *The International Journal of Quality & Reliability Management*, 20(4-5), 2003.

[82] N. R Jennings, P. Faratin, T. J Norman, P. O'Brien, and B. Odgers. Autonomous agents for business process management. *International Journal of Applied Artificial Intelligence*, 14(2):145–189, 2000.

[83] A. Josang and S. Presti. Analysing the relationship between risk and trust. In *Proceedings of the Second International Conference on Trust Management (ITrust'04)*, 2004.

[84] A. Jsang and R. Ismail. The beta reputation system. In *Proceedings of the 15th International Conference on Electronic Commerce*, 2002.

[85] Seikyung Jung, Kevin Harris, Janet Webster, and Jonathan L. Herlocker. SERF:integrating human recommendations with search. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management(CIKM'04)*, pages 571–580, New York, NY, USA, 2004. ACM Press.

[86] J. M Juran. *Quality Control Handbook*. Mcgraw-Hill, 1974.

[87] R Jurca, W Binder, and B Faltings. Reliable qos monitoring based on client feedback. In *Proceedings of the 16th International World Wide Web Conference (WWW07)*, pages 1003–1011, 2007.

[88] R Jurca and B Faltings. Reputation-based service level agreements for web services. In *Service Oriented Computing ICSOC - 2005*, Lecture Notes in Computer Science, pages 396–409. 2005.

[89] E. Kafeza and K. Karlapalem. Gaining control over time in workflow management applications. In *11th International Conference on Database and Expert Systems Applications (DEXA'00)*, pages 232–241. Springer-Verlag Heidelberg, September 2000.

[90] Sepandar D. Kamvar, Taher H. Haveliwala, Christopher D. Manning, and Gene H. Golub. Extrapolation methods for accelerating pagerank computations. In *Proceedings of the 12th international conference on World Wide Web*, pages 261–270, New York, NY, USA, 2003. ACM Press.

[91] George Karypis. Evaluation of item-based top-n recommendation algorithms. In *Proceedings of the tenth international conference on Information and knowledge management (CIKM'01)*, pages 247–254, New York, NY, USA, 2001. ACM Press.

[92] D.L. Kellogg, W.E. Youngdahl, and D.E Bowen. On the relationship between customer participation and satisfaction: two frameworks. *International Journal of Service Industry Management*, 8(3), 1997.

[93] Chan Young Kim, Jae Kyu Lee, Yoon Ho Cho, and Deok Hwan Kim. VISCORS: A visual-content recommender for the mobile web. *IEEE Intelligent Systems*, 19(6):32–39, 2004.

[94] Hyung-Jun Kim, Kyu Min Lee, Kee-Hyun Choi, and Dong-Ryeol Shin. Service discovery using FIPA-compliant ap to support scalability in ubiquitous environments. In *Proceedings of the Fourth Annual ACIS International Conference on Computer and Information Science (ICIS'05)*, pages 647–652, Washington, DC, USA, 2005. IEEE Computer Society.

[95] Jaekyeong Kim, Yoonho Cho, and Seungtae Kim. MOBICORS-Movie: A MOBIle COntents Recommender System for Movie. In *ICEB*, pages 789–794, 2004.

[96] George J. Klir and Bo Yuan. *Fuzzy sets and fuzzy logic : theory and applications*. Prentice Hall, 1995.

[97] H. Kreger. Web services conceptual architecture. http://www-4.ibm.com/software/solutions/webservices/pdf/WSCA.pdf, 2001.

[98] Heather Kreger. Fulfilling the web services promise. *Commun. ACM*, 46(6), 2003.

[99] Suang Kuo, Lulan-Yuan Lu, Chiung-Hui Huang, and Guo-Chiang Wu. Measuring users' perceived portal service quality: An empirical study. *Total Quality Management & Business Excellence*, 16(3), 2005.

[100] D. D Lamanna, J. Skene, and W. Emmerich. SLAng: a language for defining service level agreements. In *9th IEEE Workshop on Future Trends of Distributed Computing Systems*, 2003.

[101] L.A.Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965.

[102] Meehee Lee, Pyungseok Choi, and Yongtae Woo. A hybrid recommender system combining collaborative filtering with neural network. In *AH '02: Proceedings of the Second International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*, pages 531–534, London, UK, 2002. Springer-Verlag.

[103] Y. W Lee, D. M Strong, B. K Khan, and R. Y Wang. AIMQ: A methodology for information quality assessment. *Information and Management*, 40:133–146, 2002.

[104] Denis Leonard and Rodney McAdam. Total quality management in strategy and operations: dynamic grounded models. *Journal of Manufacturing Technology Management*, 15(3), 2004.

[105] T. A. Letsche and M. W. Berry. Large-scale information retrieval with latent semantic indexing. *Information Science*, pages 105–137, 1997.

[106] F. Leymann. Web services: distributed applications without limits. In *Proceedings of 10th Conference on Database Systems for Business*, pages 2–23, 2003.

[107] Baochun Li, Dongyan Xu, and Klara Nahrstedt. Towards integrated runtime solutions in qos-aware middleware. In *M3W: Proceedings of the 2001 international workshop on Multimedia middleware*, pages 11–14, New York, NY, USA, 2001. ACM Press.

[108] Lei Li and Ian Horrocks. A software framework for matchmaking based on semantic web technology. In *WWW '03: Proceedings of the 12th international*

conference on *World Wide Web*, pages 331–339, New York, NY, USA, 2003. ACM Press.

[109] Qing Li and Byeong Man Kim. Clustering approach for hybrid recommender system. In *WI '03: Proceedings of the IEEE/WIC International Conference on Web Intelligence*, pages 33–38, Washington, DC, USA, 2003. IEEE Computer Society.

[110] Qing Li, Byeong Man Kim, Dong Hai Guan, and Duk Whan Oh. A music recommender based on audio features. In *SIGIR*, pages 532–533, 2004.

[111] Qing Li, Byeong Man Kim, and Sung Hyon Myaeng. Clustering for probabilistic model estimation for CF. In *WWW '05: Special interest tracks and posters of the 14th international conference on World Wide Web*, pages 1104–1105, New York, NY, USA, 2005. ACM Press.

[112] Kurt Lichtner. An automated internet resource discovery system. In *CASCON '94: Proceedings of the 1994 conference of the Centre for Advanced Studies on Collaborative research*. IBM Press, 1994.

[113] Y. Liu, Anne H. H Ngu, and L. Zeng. QoS computation and policing in dynamic web service selection. In *The 13th international conference on World Wide Web (WWW' 04)*, pages 66–73, May 2004.

[114] H. Ludwig, A. Keller, A. Dan, and R. P King. A service level agreement language for dynamic electronic services. In *Proceedings of 4th IEEE International*

Workshop on Advanced Issues of E-Commerce and Web-based Information Systems (WECWIS'02), pages 25–32, 2002.

[115] Heiko Ludwig, Asit Dan, and Robert Kearney. Cremona: an architecture and library for creation and monitoring of ws-agreents. In *Proceedings of the 2nd international conference on Service oriented computing (ICSOC'04)*, pages 65–74, New York, NY, USA, 2004. ACM Press.

[116] Zakaria Maamar, Soraya Kouadri, and Hamdi Yahyaoui. A web services composition approach based on software agents and context. In *Proceedings of the 2004 ACM symposium on Applied computing*, pages 1619–1623. ACM Press, 2004.

[117] P. Maes, R. H Guttman, and A. G Moukas. Agents that buy and sell. *Communications of the ACM*, 42(3):81–91, 1999.

[118] Harry Mak, Irena Koprinska, and Josiah Poon. Intimate: A web-based movie recommender using text categorization. In *Proceedings of IEEE/WIC International Conference on Web Intelligence*, pages 602–605, Los Alamitos, CA, USA, 2003. IEEE Computer Society.

[119] A. Mani and A. Nagarajan. Understanding quality of service for webservices. http://www-106.ibm.com/developerworks/library/ws-quality.html, January 2002.

[120] S. Marsh. *Formalising Trust as a Computational Concept*. PhD thesis, 1994.

[121] E. M Maximilien and M. Singh. Conceptual model of web service reputation.

SIGMOD Record, ACM Special Interest Group on Management of Data, pages 36 41, 2002.

[122] E. Michael Maximilien and Munindar P. Singh. Reputation and endorsement for web services. *SIGecom Exch.*, 3(1):24 31, 2002.

[123] Sean M. McNee, Istvan Albert, Dan Cosley, Prateep Gopalkrishnan, Shyong K. Lam, Al Mamunur Rashid, Joseph A. Konstan, and John Riedl. On the recommending of citations for research papers. In *CSCW '02: Proceedings of the 2002 ACM conference on Computer supported cooperative work*, pages 116–125, New York, NY, USA, 2002. ACM Press.

[124] M. Mecella, M. Scannapieco, A. Virgillito, R. Baldoni, T. Catarci, and C. Batini. Managing data quality in cooperative information systems. In *Proceedings of 10th International Conference on Cooperative Information Systems*, pages 486–502, 2002.

[125] Daniel A. Menasce. QoS issues in web services. *IEEE Internet Computing*, 6(6):72 75, 2002.

[126] Bradley N. Miller, Istvan Albert, Shyong K. Lam, Joseph A. Konstan, and John Riedl. MovieLens unplugged: experiences with an occasionally connected recommender system. In *IUI '03: Proceedings of the 8th international conference on Intelligent user interfaces*, pages 263–266, New York, NY, USA, 2003. ACM Press.

[127] L. Mui, M. Mohtashemi, and A. Halberstadt. A computational model of trust

and reputation. In *Proceedings of 35th Hawaii International Conference on System Sciences*, pages 2423–2431, 2002.

[128] Lik Mui, M. Mohtashemi, C. Ang, P. Szolovits, and A. Halberstadt. Ratings in distributed systems: A bayesian approach. In *Workshop on Information Technologies and Systems (WITS'2001)*, 2001.

[129] Klara Nahrstedt, Hao hua Chu, and Srinivas Narayan. QoS-aware resource management for distributed multimedia applications. *Journal of High Speed Networks*, 7(3-4):229–257, 1999.

[130] Olfa Nasraoui and Mrudula Pavuluri. Accurate web recommendations based on profile-specific url-predictor neural networks. In *WWW Alt. '04: Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, pages 300–301, New York, NY, USA, 2004. ACM Press.

[131] Thanh Nguyen and Eva Tardos. Approximately maximizing efficiency and revenue in polyhedral environments. In *EC '07: Proceedings of the 8th ACM conference on Electronic commerce*, pages 11–19, New York, NY, USA, 2007. ACM Press.

[132] Tommaso Di Noia, Eugenio Di Sciascio, Francesco M. Donini, and Marina Mongiello. A system for principled matchmaking in an electronic marketplace. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 321–330, New York, NY, USA, 2003. ACM Press.

[133] T. J. Norman, A. Preece, S. Chalmers, N. R Jennings, M. Luck, V. D Dang, T. D

model of service quality and its implications for future research. *Journal of Marketing*, 49(4), 1985.

[141] A. Parasuraman, Valarie A. Zeithaml, and Leonard L. Berry. SERVQUAL: A multiple-item scale for measuring consumer perception. *Journal of Retailing*, 1988.

[142] A Parasuraman, Valarie A Zeithaml, and Arvind Malhotra. E-S-QUAL: A multiple-item scale for assessing electronic service quality. *Journal of Service Research*, 7(3):213–221, 2005.

[143] C. Patel, K. Supekar, and Y. Lee. A QoS oriented framework for adaptive management of web service based workflows. In *Database and Expert Systems Applications*, pages 826–835. Springer-Verlag Heidelberg, 2003.

[144] Resnick Paul, Zeckhauser Richard, Swanson John, and Kate Lockwood. The value of reputation on eBay: A controlled experiment page. *Experimental Economics*, 9(2):79–101, 2006.

[145] Michael Pazzani and Daniel Billsus. Learning and revising user profiles: The identification of interesting web sites. *Machine Learning*, 27(3):313–331, 1997.

[146] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Network of Plausible Inference*. Morgan Kaufmann Publishers Inc, 1988.

[147] Saverio Perugini, Marcos A. Gonalves, and Edward A. Fox. Recommender systems research: A connection-centric survey. *Journal of Intelligent Information Systems*, 23(2):107–143, 2004.

[148] S. A Petersen and M. Gruninger. An agent-based model to support the formation of virtual enterprises. In *International ICSC Symposium on Mobile Agents and Multi-agents in Virtual Organisations and E-Commerce*, 2000.

[149] G. Piccinelli, A. Finkelstein, and S. L Williams. Service-oriented workflow: the dysco framework. In *Proceedings of the 29th EUROMICRO Conference New Waves in System Architecture (EUROMICRO03)*, pages 291 – 297, September 2003.

[150] G. Piccinelli and E. Stammers. From e-processes to e-networks: an e-service-oriented approach. In *Proceedings of 3rd International Conference on Internet Computing*, pages 549–553, 2002.

[151] Shuping Ran. A model for web services discovery with QoS. *SIGecom Exch.*, 4(1):1–10, 2003.

[152] Olga Ratsimor, Dipanjan Chakraborty, Anupam Joshi, Timothy Finin, and Yelena Yesha. Service discovery in agent-based pervasive computing environments. volume 9, pages 679–692, Hingham, MA, USA, 2004. Kluwer Academic Publishers.

[153] Kevin Regan, Pascal Poupart, and Robin Cohen. Bayesian reputation modeling in e-marketplaces sensitive to subjectivity, deception and change. In *AAAI*, 2006.

[154] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. GroupLens: an open architecture for collaborative filtering of netnews.

In *CSCW '94: Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186, New York, NY, USA, 1994. ACM Press.

[155] Paul Resnick and Richard Zeckhauser. Trust among strangers in internet transactions: Empirical analysis of eBay's reputation system. *The Economics of the Internet and E-Commerce*, 2002.

[156] V. Robles, P. Larranaga, E. Menasalvas, M. S. Perez, and V. Herves. Improvement of naive bayes collaborative filtering using interval estimation. In *WI '03: Proceedings of the IEEE/WIC International Conference on Web Intelligence*, pages 168–174, Washington, DC, USA, 2003. IEEE Computer Society.

[157] H. Roest and R. Pieters. The nomological net of perceived service quality. *International Journal of Service Industry Management*, 8(4), 1997.

[158] Paat Rusmevichientong and David P. Williamson. An adaptive algorithm for selecting profitable keywords for search-based advertising services. In *EC '06: Proceedings of the 7th ACM conference on Electronic commerce*, pages 260–269, New York, NY, USA, 2006. ACM Press.

[159] R. T Rust and P. K Kannan. E-service: a new paradigm for business in the electronic environment. *Communications of the ACM*, 46(6):36–42, 2003.

[160] R. T. Rust and A. J. Zahorik. Return on quality(ROQ): Making service quality financially accountable. *Journal of Marketing*, 1995.

[161] Arthur L Rutledge, Kenneth R Tillery, Bryan Kethley, and Kiran J Desai. The treatment of quality in us production and operations management textbooks:

A reassessment and extension ten years after. *The International Journal of Quality & Reliability Management*, 21(4-5), 2004.

[162] Jordi Sabater and Carles Sierra. Reputation and social network analysis in multi-agent systems. In *AAMAS '02: Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, pages 475–482, New York, NY, USA, 2002. ACM Press.

[163] Jordi Sabater and Carles Sierra. Social ReGreT, a reputation model based on social relations. *SIGecom Exch.*, 3(1):44–56, 2002.

[164] M. Schillo, P. Funk, and M. Rovatsos. Using trust for detecting deceitful agents in artificial societies. *Applied Artificial Intelligence, Special Issue on Trust, Deception and Fraud in Agent Societies*, pages 825–848, 2000.

[165] H. Schuster, D. Georgakopoulos, A. Cichocki, and D. Baker. Modeling and composing service-based and reference process-based multi-enterprise processes. In *12th International Conference on Advanced Information Systems Engineering (CAISE 2000)*, pages 247–263. Springer-Verlag Heidelberg, June 2000.

[166] Ali ShaikhAli, Omer F. Rana, Rashid Al-Ali, and David W. Walker. UD-DIe: An extended registry for web services. In *SAINT-W '03: Proceedings of the 2003 Symposium on Applications and the Internet Workshops (SAINT'03 Workshops)*, pages 85–89, Washington, DC, USA, 2003. IEEE Computer Society.

[167] Upendra Shardanand and Pattie Maes. Social information filtering: algorithms

for automating word of mouth. In *CHI '95: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 210–217, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co.

[168] D.J. Shemwell, U. Yavas, and Z. Bilgin. Customer-service provider relationships: an empirical test of a model of service quality, satisfaction and relationship oriented outcome. *International Journal of Service Industry Management*, 9(2):155–168, 1998.

[169] Bradley Simmons and Hanan Lutfiyya. Policies, grids and autonomic computing. In *DEAS '05: Proceedings of the 2005 workshop on Design and evolution of autonomic application software*, pages 1–5, New York, NY, USA, 2005. ACM Press.

[170] Noel Yee-Man Siu and Jeremy Chi-Wah Mou. Measuring service quality in internet banking: The case of hong kong. *Journal of International Consumer Marketing*, 17(4), 2005.

[171] J. Skene, D. Lamanna, and W. Emmerich. Precise service level agreements. In *26th International Conference on Software Engineering*, pages 179 – 188, May 2004.

[172] Ian Soboroff and Charles Nicholas. Collaborative filtering and the generalized vector space model (poster session). In *SIGIR '00: Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 351–353, New York, NY, USA, 2000. ACM Press.

[173] M. Soderlund. Customer satisfaction and its consequences on customer behaviour revisited - the impact of different levels of satisfaction on word-of-mouth, feedback to the supplier and loyalty. *International Journal of Service Industry Management*, 9(2), 1998.

[174] Monika Solanki, Antonio Cau, and Hussein Zedan. Augmenting semantic web service descriptions with compositional specification. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, pages 544–552, New York, NY, USA, 2004. ACM Press.

[175] Ji Hee Song and Jason Q Zhang. Why do people shop online?: Exploring the quality of online shopping experience. *American Marketing Association*, 15, 2004.

[176] Rui Sousa and Christopher A Voss. Quality management re-visited: A reflective review and agenda for future research. *Journal of Operations Management*, 20(1), 2002.

[177] Natenapa Sriharee, Twittie Senivongse, Kunal Verma, and Amit Sheth. On using WS-Policy, Ontology, and rule reasoning to discover web services. In *Intelligence in Communication Systems*, 2004.

[178] Patrick J. Stockreisser, Jianhua Shao, W. Alex Gray, and Nick J. Fiddian. Supporting qos monitoring in virtual organisations. In *4th International Conference on Service Oriented Computing (ICSOC)*, pages 447–452, 2006.

[179] R Sumra and D Arulazi. Quality of service for web servicesdemystification, limitations, and best practices. http://www.developer.com, 2004.

[180] Bhushan Shankar Suryavansh, Nematollaah Shiri, and Sudhir P. Mudur. Improving the effectiveness of model based recommender systems for highly sparse and noisy web usage data. In *Web Intelligence*, pages 618–621, Los Alamitos, CA, USA, 2005. IEEE Computer Society.

[181] A Talwar, R Jurca, and B Faltings. Understanding user behavior in online feedback reporting. In *Proceedings of the ACM Conference on Electronic Commerce EC'07*, pages 134–142, 2007.

[182] Khopkar Tapan, Li Xin, and Resnick Paul. Self-selection, slipping, salvaging, slacking, and stoning: the impacts of negative feedback at eBay. In *Proceedings of ACM EC 05 Conference on Electronic Commerce*, pages 223–231, 2005.

[183] W. T. Luke Teacy, Jigar Patel, Nicholas R. Jennings, and Michael Luck. Coping with inaccurate reputation sources: experimental analysis of a probabilistic trust model. In *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 997–1004, New York, NY, USA, 2005. ACM Press.

[184] CONOISE Team. www.conoise.org. URL, 2006.

[185] V. Tosic, B. Pagurek, K. Patel, B. Esfandiari, and W. Ma. Management applications of the Web Service Offerings Language (WSOL). In *15th International*

Conference on Advanced Information Systems Engineering,CAiSE 2003, pages 468 – 484. Springer-Verlag Heidelberg, 2003.

[186] David Trastour, Claudio Bartolini, and Chris Preist. Semantic web support for the business-to-business e-commerce lifecycle. In *WWW '02: Proceedings of the 11th international conference on World Wide Web*, pages 89–98, New York, NY, USA, 2002. ACM Press.

[187] K. Trzec and D. Huljenic. Intelligent agents for QoS management. In *Proceedings of First International Conference on Autonomous Agents and Multi Agent Systems*, pages 1405–1412, 2002.

[188] B. W Tuchman. The decline of quality. *New York Times Magazine*, November(2):38–41, 1980.

[189] R.W.E. van der Wal, A. Pampallis, and C. Bond. Service quality in a cellular telecommunications company: A south african experience. *Managing Service Quality*, 12(5), 2002.

[190] D. Veit, J. P. Mller, M. Schneider, and B. Fiehn. Matchmaking for autonomous agents in electronic marketplaces. In *AGENTS '01: Proceedings of the fifth international conference on Autonomous agents*, pages 65–66, New York, NY, USA, 2001. ACM Press.

[191] Nalini Venkatasubramanian, Carolyn Talcott, and Gul A. Agha. A formal model for reasoning about adaptive QoS-enabled middleware. *ACM Transactions on Software Engineering and Methodology*, 13(1):86–147, 2004.

[192] W3Schools. WSDL and UDDI. www.w3schools.com/wsdl, September 2006.

[193] Guijun Wang, Alice Chen, Changzhou Wang, Casey Fung, and Stephen Uczekaj. Integrated quality of service (QoS) management in service-oriented enterprise architectures. In *EDOC '04: Proceedings of the Enterprise Distributed Object Computing Conference, Eighth IEEE International (EDOC'04)*, pages 21-32, Washington, DC, USA, 2004. IEEE Computer Society.

[194] P.Y. Wang, Yechiam Yemini, D. Florissi, and P. Florissi. QoSME: toward QoS management and guarantees. In *Proceedings of the International Conference on Communication Technology*, pages 868 - 875, 2000.

[195] R. Y Wang and D. M Strong. Beyond accuracy: what data quality means to data consumers. *Journal of Management Information Systems*, 12(4):5-34, 1996.

[196] Xiaoling Wang, Kun Yue, Joshua Zhexue Huang, and Aoying Zhou. Service selection in dynamic demand-driven web services. In *IEEE International Conference on Web Services (ICWS'04)*, pages 376 - 383, 2004.

[197] Y. Wang and J. Vassileva. Bayesian network-based trust model. In *Proceedings. IEEE/WIC International Conference on Web Intelligence, 2003. WI 2003.*, pages 372 - 378, 2003.

[198] Yi-Shun Wang and Tzung-I Tang. Assessing customer perceptions of website service quality in digital marketing environments. *Journal of End User Computing*, 15(3), 2003.

[199] Yonggui Wang, Hing-Po Lo, and Yongheng Yang. An integrated framework for service quality, customer value, satisfaction: Evidence from china's telecommunication industry. *Information Systems Frontiers*, 6(4), 2004.

[200] Andrew Whitby, Audun Josang, and Jadwiga Indulska. Filtering out unfair ratings in bayesian reputation systems. In *Proceedings of the 7th Int Workshop on Trust in Agent Societies*, 2004.

[201] Wikipedia. http://en.wikipedia.org/wiki/framerate. WWW, January 2007.

[202] Eric Wohlstadter, Stefan Tai, Thomas Mikalsen, Isabelle Rouvellou, and Premkumar Devanbu. GlueQoS: Middleware to sweeten Quality-of-Service policy interactions. In *ICSE '04: Proceedings of the 26th International Conference on Software Engineering*, pages 189–199, Washington, DC, USA, 2004. IEEE Computer Society.

[203] Luo Xiao, Dieter Wissmann, Michael Brown, and Stephan Jablonski. Information extraction from the web: System and techniques. *Applied Intelligence*, 21(2):195–224, 2004.

[204] Li Xiong and Ling Liu. A reputation-based trust model for peer-to-peer ecommerce communities [extended abstract]. In *Proceedings of the 4th ACM conference on Electronic commerce*, pages 228–229, New York, NY, USA, 2003. ACM Press.

[205] D. Xu, K. Nahrstedt, and D. Wichadakul. QoS-aware discovery of wide-area distributed services.

[206] Gui-Rong Xue, Chenxi Lin, Qiang Yang, WenSi Xi, Hua-Jun Zeng, Yong Yu, and Zheng Chen. Scalable collaborative filtering using cluster-based smoothing. In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 114–121, New York, NY, USA, 2005. ACM Press.

[207] Fengzhao Yang, Yangyong Zhu, and Bole Shi. A new algorithm for performing ratings-based collaborative filtering. In *Proceedings of 5th Asia-Pacific Web Conference on Web Technologies and Applications*, 2003.

[208] Bin Yu and Munindar P. Singh. An evidential model of distributed reputation management. In *Proceedings of First International Conference on Autonomous Agents and Multi Agent Systems*, pages 294–301, 2002.

[209] Bin Yu and Munindar P. Singh. Searching social networks. In *AAMAS '03: Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 65–72, New York, NY, USA, 2003. ACM Press.

[210] K. Yu, X. Xu, M. Ester, and H.P. Kriegel. Feature weighting and instance selection for collaborative filtering: An information-theoretic approach. *Knowledge and Information Systems: An International Journal*, 2003.

[211] Kai Yu, Volker Tresp, and Shipeng Yu. A nonparametric hierarchical bayesian framework for information filtering. In *SIGIR '04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 353–360, New York, NY, USA, 2004. ACM Press.

[212] G. Zacharia, A. Moukas, and P. Maes. Collaborative reputation mechanisms in electronic marketplaces. In *Proceedings of the 32nd Annual Hawaii International Conference on System Sciences (HICSS-32)*, 1999.

[213] L. A Zadeh. Similarity relations and fuzzy orderings. *Fuzzy Sets and Applications*, pages 81–104, 1997.

[214] Milan Zeleny. *Multiple Criteria Decision Making*. McGraw Hill, 1981.

[215] L. Zeng, B. Benatallah, M. Dumas, and J. Kalagnanam. Quality driven web service composition. In *Proceedings of Twelfth International Conference on World Wide Web*, pages 411–421, 2003.

[216] L. Zeng, B. Benatallah, Anne H.H Ngu, M. Dumas, J. Kalagnanam, and H. Chang. QoS-aware middleware for web services composition. *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING*, 30(5):311–327, May 2004.

[217] Zili Zhang and Chengqi Zhang. An improvement to matchmaking algorithms for middle agents. In *AAMAS '02: Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, pages 1340–1347, New York, NY, USA, 2002. ACM Press.

[218] C. Zhou, L. T Chia, and B. S Lee. DAML-QoS Ontology for Web services. In *Proceedings of the IEEE International Conference on Web Services*, 2004.

[219] Justin Zobel and Alistair Moffat. Inverted files for text search engines. *ACM Computing Survey*, 38(2), 2006.

Appendix A

Media Application Schema

This Appendix shows the *media* application schema.

```
1  <?xml version="1.0" encoding="ISO-8859-1" ?>

   <!DOCTYPE rdf:RDF>

3  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"

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

5    xmlns:service="http://www.cs.cf.ac.uk/research/conoise/jan05/

        ontologies/service#"

     xmlns:profile="http://www.cs.cf.ac.uk/research/conoise/jan05/

        ontologies/profile#"

7    xmlns:quality="http://www.cs.cf.ac.uk/research/conoise/jan05/

        ontologies/quality#"

     xmlns:package="http://www.cs.cf.ac.uk/research/conoise/jan05/

        ontologies/package#"

9    xmlns:product="http://www.cs.cf.ac.uk/research/conoise/jan05/
```

```
            ontologies/product#"

      xmlns:media="http://www.cs.cf.ac.uk/research/conoise/jan05/

            ontologies/media#"

11    xml:base="http://www.cs.cf.ac.uk/research/conoise/jan05/

            ontologies/media#">                                    •


13  <rdfs:Class rdf:about="http://www.cs.cf.ac.uk/research/conoise/

          jan05/ontologies/media#Media">

        <rdfs:subClassOf rdf:resource="http://www.cs.cf.ac.uk/research/

              conoise/jan05/ontologies/product#Product" />

15  </rdfs:Class>


17  <rdfs:Class rdf:about="http://www.cs.cf.ac.uk/research/conoise/

          jan05/ontologies/media#Communication">

        <rdfs:subClassOf rdf:resource="http://www.cs.cf.ac.uk/research/

              conoise/jan05/ontologies/media#Media" />

19  </rdfs:Class>


21  <rdfs:Class rdf:about="http://www.cs.cf.ac.uk/research/conoise/

          jan05/ontologies/media#PhoneCalls">

        <rdfs:subClassOf rdf:resource="http://www.cs.cf.ac.uk/research/

              conoise/jan05/ontologies/media#Communication" />

23  </rdfs:Class>
```

```
25 <rdfs:Class rdf:about="http://www.cs.cf.ac.uk/research/conoise/
      jan05/ontologies/media#TextMessaging">
    <rdfs:subClassOf rdf:resource="http://www.cs.cf.ac.uk/research/
        conoise/jan05/ontologies/media#Communication" />
27 </rdfs:Class>


29 <rdfs:Class rdf:about="http://www.cs.cf.ac.uk/research/conoise/
      jan05/ontologies/media#Multimedia">
    <rdfs:subClassOf rdf:resource="http://www.cs.cf.ac.uk/research/
        conoise/jan05/ontologies/media#Media" />
31 </rdfs:Class>


33 <rdfs:Class rdf:about="http://www.cs.cf.ac.uk/research/conoise/
      jan05/ontologies/media#HtmlContent">
    <rdfs:subClassOf rdf:resource="http://www.cs.cf.ac.uk/research/
        conoise/jan05/ontologies/media#Multimedia" />
35 </rdfs:Class>


37 <rdfs:Class rdf:about="http://www.cs.cf.ac.uk/research/conoise/
      jan05/ontologies/media#VideoContent">
    <rdfs:subClassOf rdf:resource="http://www.cs.cf.ac.uk/research/
        conoise/jan05/ontologies/media#Multimedia" />
39 </rdfs:Class>
```

```
41  <rdfs:Class rdf:about="http://www.cs.cf.ac.uk/research/conoise/
        jan05/ontologies/media#MovieContent">
      <rdfs:subClassOf rdf:resource="http://www.cs.cf.ac.uk/research/
          conoise/jan05/ontologies/media#VideoContent" />
43  </rdfs:Class>


45  <rdf:Property rdf:about="http://www.cs.cf.ac.uk/research/conoise/
        jan05/ontologies/media#htmlContentType">
      <rdfs:domain rdf:resource="http://www.cs.cf.ac.uk/research/
          conoise/jan05/ontologies/media#HtmlContent" />
47    <rdfs:range rdf:resource="http://www.cs.cf.ac.uk/research/
          conoise/jan05/ontologies/media#MediaCategory" />
      </rdf:Property>

49
    <rdf:Property rdf:about="http://www.cs.cf.ac.uk/research/conoise/
        jan05/ontologies/media#mediaDescription">
51    <rdfs:domain rdf:resource="http://www.cs.cf.ac.uk/research/
          conoise/jan05/ontologies/media#Multimedia" />
      <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#
          Literal" />
53  </rdf:Property>


55  <rdf:Property rdf:about="http://www.cs.cf.ac.uk/research/conoise/
        jan05/ontologies/media#mediaStyle">
```

```
        <rdfs:domain rdf:resource="http://www.cs.cf.ac.uk/research/
            conoise/jan05/ontologies/media#VideoContent" />
57      <rdfs:range rdf:resource="http://www.cs.cf.ac.uk/research/
            conoise/jan05/ontologies/media#MediaCategory" />
    </rdf:Property>

59

    <rdf:Property rdf:about="http://www.cs.cf.ac.uk/research/conoise/
        jan05/ontologies/media#numberOfMovies">
61      <rdfs:domain rdf:resource="http://www.cs.cf.ac.uk/research/
            conoise/jan05/ontologies/media#MovieContent" />
        <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#
            Literal" />
63  </rdf:Property>


65  <rdf:Property rdf:about="http://www.cs.cf.ac.uk/research/conoise/
        jan05/ontologies/media#numberOfMessages">
        <rdfs:domain rdf:resource="http://www.cs.cf.ac.uk/research/
            conoise/jan05/ontologies/media#TextMessaging" />
67      <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#
            Literal" />
    </rdf:Property>

69

    <rdf:Property rdf:about="http://www.cs.cf.ac.uk/research/conoise/
        jan05/ontologies/media#numberOfMinutes">
```

```
71    <rdfs:domain rdf:resource="http://www.cs.cf.ac.uk/research/
          conoise/jan05/ontologies/media#PhoneCalls" />
      <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#
          Literal" />
73 </rdf:Property>


75 <rdf:Property rdf:about="http://www.cs.cf.ac.uk/research/conoise/
      jan05/ontologies/media#subscriptionType">
      <rdfs:domain rdf:resource="http://www.cs.cf.ac.uk/research/
          conoise/jan05/ontologies/product#Product" />
77    <rdfs:range rdf:resource="http://www.cs.cf.ac.uk/research/
          conoise/jan05/ontologies/media#SubscriptionCategory" />
      </rdf:Property>

79

   <rdf:Property rdf:about="http://www.cs.cf.ac.uk/research/conoise/
      jan05/ontologies/media#updateFrequency">
81    <rdfs:domain rdf:resource="http://www.cs.cf.ac.uk/research/
          conoise/jan05/ontologies/media#HtmlContent" />
      <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#
          Literal" />
83 </rdf:Property>


85 <rdf:Property rdf:about="http://www.cs.cf.ac.uk/research/conoise/
      jan05/ontologies/media#bandwidth">
```

```
       <rdfs:domain rdf:resource="http://www.cs.cf.ac.uk/research/
          conoise/jan05/ontologies/media#Multimedia" />
87     <rdfs:range rdf:resource="http://www.cs.cf.ac.uk/research/
          conoise/jan05/ontologies/media#BandwidthCategory" />
     </rdf:Property>

89

     <rdf:Property rdf:about="http://www.cs.cf.ac.uk/research/conoise/
        jan05/ontologies/media#frameRate">
91     <rdfs:domain rdf:resource="http://www.cs.cf.ac.uk/research/
          conoise/jan05/ontologies/media#VideoContent" />
       <rdfs:range rdf:resource="http://www.cs.cf.ac.uk/research/
          conoise/jan05/ontologies/media#FrameRateCategory" />
93   </rdf:Property>


95   <rdf:Property rdf:about="http://www.cs.cf.ac.uk/research/conoise/
        jan05/ontologies/media#soundQuality">
       <rdfs:domain rdf:resource="http://www.cs.cf.ac.uk/research/
          conoise/jan05/ontologies/media#VideoContent" />
97     <rdfs:range rdf:resource="http://www.cs.cf.ac.uk/research/
          conoise/jan05/ontologies/media#SoundQualityCategory" />
     </rdf:Property>

99

     <rdfs:Class rdf:about="http://www.cs.cf.ac.uk/research/conoise/
        jan05/ontologies/media#MediaCategory">
```

```
101 <rdfs:subClassOf rdf:resource="http://www.w3.org/2000/01/rdf-
        schema#Resource" />
    </rdfs:Class>

103

    <rdfs:Class rdf:about="http://www.cs.cf.ac.uk/research/conoise/
        jan05/ontologies/media#SubscriptionCategory">
105   <rdfs:subClassOf rdf:resource="http://www.w3.org/2000/01/rdf-
          schema#Resource" />
    </rdfs:Class>

107

    <rdfs:Class rdf:about="http://www.cs.cf.ac.uk/research/conoise/
        jan05/ontologies/media#BandwidthCategory">
109   <rdfs:subClassOf rdf:resource="http://www.cs.cf.ac.uk/research/
          conoise/jan05/ontologies/quality#Quality" />
    </rdfs:Class>

111

    <rdfs:Class rdf:about="http://www.cs.cf.ac.uk/research/conoise/
        jan05/ontologies/media#FrameRateCategory">
113   <rdfs:subClassOf rdf:resource="http://www.cs.cf.ac.uk/research/
          conoise/jan05/ontologies/quality#Quality" />
    </rdfs:Class>

115

    <rdfs:Class rdf:about="http://www.cs.cf.ac.uk/research/conoise/
        jan05/ontologies/media#SoundQualityCategory">
```

```
117    <rdfs:subClassOf rdf:resource="http://www.cs.cf.ac.uk/research/
           conoise/jan05/ontologies/quality#Quality" />
       </rdfs:Class>

119

       <!-- sample "constant" terms

121    -->


123    <media:MediaCategory rdf:ID="http://www.cs.cf.ac.uk/research/
           conoise/jan05/ontologies/media#scienceFiction" />


125    <media:MediaCategory rdf:ID="http://www.cs.cf.ac.uk/research/
           conoise/jan05/ontologies/media#horror" />


127    <media:MediaCategory rdf:ID="http://www.cs.cf.ac.uk/research/
           conoise/jan05/ontologies/media#comedy" />


129    <media:MediaCategory rdf:ID="http://www.cs.cf.ac.uk/research/
           conoise/jan05/ontologies/media#drama" />


131    <media:MediaCategory rdf:ID="http://www.cs.cf.ac.uk/research/
           conoise/jan05/ontologies/media#documentary" />


133    <media:MediaCategory rdf:ID="http://www.cs.cf.ac.uk/research/
           conoise/jan05/ontologies/media#news" />
```

```
135  <media:MediaCategory rdf:ID="http://www.cs.cf.ac.uk/research/
         conoise/jan05/ontologies/media#sport" />

137  <media:SubscriptionCategory rdf:ID="http://www.cs.cf.ac.uk/
         research/conoise/jan05/ontologies/media#daily" />

139  <media:SubscriptionCategory rdf:ID="http://www.cs.cf.ac.uk/
         research/conoise/jan05/ontologies/media#weekly" />

141  <media:SubscriptionCategory rdf:ID="http://www.cs.cf.ac.uk/
         research/conoise/jan05/ontologies/media#monthly" />

143  <media:SubscriptionCategory rdf:ID="http://www.cs.cf.ac.uk/
         research/conoise/jan05/ontologies/media#quarterly" />

145  <media:SubscriptionCategory rdf:ID="http://www.cs.cf.ac.uk/
         research/conoise/jan05/ontologies/media#biannual" />

147  <media:SubscriptionCategory rdf:ID="http://www.cs.cf.ac.uk/
         research/conoise/jan05/ontologies/media#annual" />

149  <media:BandwidthCategory rdf:ID="kbps1024" />
```

```
151  <media:BandwidthCategory  rdf:ID="kbps56"  />

153  <media:BandwidthCategory  rdf:ID="kbps44"  />

155  <media:FrameRateCategory  rdf:ID="fps24"  />

157  <media:FrameRateCategory  rdf:ID="fps12"  />

159  <media:FrameRateCategory  rdf:ID="fps6"  />

161  <media:SoundQualityCategory  rdf:ID="dolby5_1"  />

163  <media:SoundQualityCategory  rdf:ID="mono"  />

165  </rdf:RDF>
```