A Quality-aware Cloud Selection Service for Computational Modellers

Shahzad Ahmed Nizamani

Submitted in accordance with the requirements for the degree of
Doctorate of Philosophy

The University of Leeds
School of Computing

July, 2012

The candidate confirms that the work submitted is his own, except where work which has formed part of jointly-authored publications has been included. The contribution of the candidate and the other authors to this work has been explicitly indicated below. The candidate confirms that appropriate credit has been given within the thesis where reference has been made to the work of others.

This copy has been supplied on the understanding that it is copyright material and that no quotation from the thesis may be published without proper acknowledgement.

© 2012 The University of Leeds and Shahzad Ahmed Nizamani

Acknowledgements

I would like to start by expressing my gratitude to both my supervisors Professor Peter Dew and Dr. Karim Djemame. It has been a genuine pleasure working with them. I would like to thank you again for your energy, commitment, interest and support throughout the course of my studies.

I am also grateful to *Professor Peter Jimmack* and *Sarfaraz* for proving a user's perspective on the research and for providing information on the computational modelling; and to *Dr Brandon Bennet* for his input on the ontology.

Thanks to all the people in the *Collaborative Systems and Performance Research Group* at the School of Computing, University of Leeds who informed and influenced my work. I wish to offer special thanks to all my friends in Leeds specially those in Lab 7.14 for making my days more interesting and entertaining.

I would like to take this opportunity let my family formally know that I admire their support very much. To my father *Sher Ahmed* thank you for your constant belief in me regardless of the challenge, to my mother *Zebunissa* thank you for choosing my education over my company, to my wife *Rabia* thank you for your patience and support, to my daughter *Eliza* thank you for bringing a smile to my face every day and not forgetting my sister *Sanobar* thank you for your encouragement. I would also like to thank my grandparents; not forgetting the many uncles and aunts thank you everyone.

Last and by no means least I would like to thank Mehran University of Engineering & Technology and Higher Education Commission Pakistan; without funding it would have never happened.

Publications

The research presented in chapters 4 to 6; describing the architecture, implementation and evaluation of the QaComPS has been summarized and presented in the following publications.

[to appear] Shahzad Nizamani; Peter Dew; Karim Djemame *A Quality-aware Cloud Selection Service for Computational Modellers.* **International Journal of Cloud Computing. 2013: Inderscience.**

Peter Dew; Shahzad Nizamani; QAComPS: A Quality-aware Federated Computational Semantic Web Service for Computational Modellers. **Semantic Web and Web Services. 2011:World Comp.**

Abstract

This research sets out to help computational modellers, to select the most cost effective Cloud service provider. This is when they opt to use Cloud computing in preference to using the in-house High Performance Computing (HPC) facilities.

A novel *Quality-aware computational Cloud Selection (QAComPS)* service is proposed and evaluated. This selects the best (cheapest) Cloud provider's service. After selection it automatically sets-up and runs the selected service. QaComPS includes an integrated ontology that makes use of OWL 2 features. The ontology provides a standard specification and a common vocabulary for describing different Cloud provider's services. The semantic descriptions are processed by the QaComPS Information Management service. These provider descriptions are then used by a filter and the MatchMaker to automatically select the highest ranked service that meets the user's requirements. A SAWSDL interface is used to transfer semantic information to/from the QAComPS Information Management service and the non semantic selection and run services.

QAComPS selection service has been quantitatively evaluated for accuracy and efficiency against Quality Matchmaking Process (QMP) and Analytical Hierarchy Process (AHP). The service was also evaluated qualitatively by a group of computational modellers. The results for the evaluation were very promising and demonstrated QaComPS's potential to make Cloud computing more accessible and cost effective for computational modellers.

Table of Contents

List of Tables

List of Figures

Abbreviations

AMI: Amazon Machine Image

API: Application Programming Interface

EC2: Elastic Compute Cloud

HPC: High Performance Computing

IaaS: Infrastructure as a Service

MCDA: Multi Criteria Decision Analysis

NIST: National Institute of Science and Technology

OS: Operating System

OWL: Web Ontology Language

PaaS: Platform as a Service

PoF: Probability of Failure

QaComPS: Quality aware Computational Cloud Selection

QMP: Quality Matchmaking Process

QoS: Quality of Service

RBAC: Roll Back Access Control

RDF: Resource Description Framework

RDFS: RDF Schema

SaaS: Software as a Service

SAWSDL: Semantic Annotations for WSDL

SLA: Service Level Agreement

SOAP:	Simple Object Access Protocol
SPARQL:	SPARQL Protocol and RDF Query Language
SQL:	Structured Query Language
SWRL:	Semantic Web Rule Language
SWS:	Semantic Web Service
URI:	Universal Resource Identifier
URL:	Universal Resource Locator
VM:	Virtual Machine
WAH:	Web Application Hosting
W3C:	World Wide web Consortium
WSDL:	Web Service Description Language
XML:	eXtensible Markup Language

Chapter 1
Introduction

Computational modellers (also referred to as users) address complex, real-world problems through building computerised models of physical phenomena. They operate in the fields of physical, financial management and life sciences and need access to High Performance Computing (HPC). Typically their requirements were fulfilled by the in-house HPC machines. The benefits of the in-house HPC include resource ownership which enables trust and security as it is a closed system. However the computational modellers have experienced frustrations; as according to a leading computational Professor:

"***Computational modellers have been experiencing frustrations in two areas. Firstly, they feel disadvantaged by the way local HPC facilities schedule jobs. The turnaround time of these jobs can be unpredictable depending on the size of the HPC job queue. Large jobs suffer the most as these are limited to weekends. Secondly, the high cost of reliably running the service. The maintenance costs are inflexible and do not cater very well for their computational service needs, as they vary throughout the model development process."***

Due to this computational modellers have looked for alternatives such as Cloud computing (Peter Mell 2011). Cloud computing has evolved over the last five years from a hype to a market standard. According to National Institute of Standards and Technology (NIST) "Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction". Accessing Cloud resources is simple as all a user needs is Internet access and a credit card. As Cloud is on-demand a user can log-on at any time and acquire any amount of resources.

Cloud providers offer the user a choice of different Virtual Machines (VMs); where a VM emulates a physical machine. This emulation is performed by hardware virtualization where a physical machine is used for creating VMs. Each VM has processor, memory, storage and other resources. Price of a VM depends on the allocated resources (e.g. the amount of run-time memory and the number of CPU cycles). The pricing for the VMs is pay per use as users are charged per hour for the VMs and there are no membership fees. This is an advantage for organizations (e.g. Universities) as they do not have to make lumping, large capital investments thus improving their cash flow. Computational modellers benefit because there are more on-demand service options than the "one-size-fits-all" service provided by the in-house HPC facilities. However this does pose a challenge as more than one provider's can fulfil the user's resource requirements. In such cases Quality of Service (QoS) becomes the service differentiating criteria (Tran, Tsuji et al. 2009). QoS is the rating of the provider's progress in terms of reliability, security, and many other quality parameters.

It is very difficult to enforce QoS guarantees without human involvement so the current public Cloud providers offer only "best-efforts". This is referred to as *Quality-awareness*. The Cloud providers can:

- Drop the service at any time in cases such as overload. This is critical in terms of cost as the provider would reimburse only for the duration of the failure. For example if a job, which is running for 10 hours has to be restarted due to a five minute failure. The user would be reimbursed for only five minutes. In this case the user not only loses money but also time as the job needs to be re-run.

- Providers offer no guarantees concerning the response time, job throughput etc.). For example see the Amazon EC2 service level agreement (SLA) (Amazon 2012). Rochwerger et.al (Rochwerger 2010) argue that cloud providers have only recently begun to address the requirements of enterprise solutions, such as support for infrastructure service-level agreements.

Running large HPC jobs; on either local HPC or the Cloud is very expensive, so QoS is likely to be an important concern for the users as it contains information such as the probability of service failure. This is particularly important for users who wish to run large computational jobs particularly if they have to meet strict time constraints.

The research problem is to provide a service that can mediate across a number of computational Cloud providers, to select and run (transparently) the best (e.g. cheapest) Cloud provider's VM subject to user's requirements. A novel Quality-aware Computational Cloud Selection (QAComPS) service is proposed to address the aforementioned issues. Its main features are:

(1) A Cloud providers service ontology to integrate the information on the QoS and Cloud provider's resources with associated costs;

(2) An automatic selection process to discover the best VM that meets the computational modellers QoS and resources requirements;

(3) A semantic annotation for Web service description language (SAWSDL) interface between the semantic Information Management service and the non

semantic selection, run services. The interface would be used for querying and updating the QoS information.

The QaComPS service has three main elements: (1) Service ontology which provides a consistent semantic data model for describing QoS metrics that are non-functional properties; (2) Matchmaker to rank the Web services using a Multi Criteria Decision Analysis (MCDA) algorithm with a predefined criterion; and (3) a component for setting up and running the selected provider's VM.

1.1 Research Aims & Objectives

The aim of this research is to improve the ability of mathematical modellers in general and computational modellers in particular, enabling them to discover and utilize Cloud resources effectively. The key objectives for the research are as follows:

- **Explore user's resource requirements**

  Computational modeller's resources requirements needed to be identified in order to purse this research. These would be collected by conducting interviews with the experts. The interviews were recorded and analysed for user requirements. An in-depth literature review would be useful in identifying the technical requirements.

- **Design and develop a service to facilitate the user in accessing Cloud resources**

  This objective concerns the main deliverable of the research i.e. QaComPS. The service design included an Information Management service (internal) which was responsible for storing and updating user's and Cloud provider's information. This was used by the selection service for processing user's queries.

- **Evaluate the service for efficiency and accuracy**

  To a user, an accurate response in a timely fashion is of prime importance. A matchmaker processes user's requests and identifies a suitable solution. In an effort to improve the efficiency of the matchmaking process this research introduces a novel approach to matchmaking. The MatchMaker would be evaluated for efficiency and accuracy, against two other matchmakers namely QMP (ELEYAN, Amna et al. 2004) and AHP (Haas and Meixner 2005) .

- **Single log-on access to a broad set of Cloud providers**

  The number of public and private Cloud providers is on the rise. Locating a provider and identifying the services it offer takes time as for every provider a separate user account is required. Creating a user account requires the user to share his credit/debit card. QaComPS enables the user to create a single account for accessing a number of participating Cloud providers.

- **Single vocabulary for Cloud providers**

  Cloud providers do not share a common vocabulary and use different terms to describe the same thing this is resolved by the QaComPS as it has a single set of terms to describe the services offered by providers.

1.2 Research Methodology

The research methodology is composed of processes, methods and tools. System development, quantitative and qualitative methodologies were used to achieve the objectives of this research.

- **System development methodology** (Vidgen 2002) was used for developing the envisioned system. System development was undertaken using iterative and incremental process. This involves feedback loops for improving the solution. The system development includes identifying the user requirements, developing the system architecture, designing the system, implementing the design and evaluating the prototype.

- **Quantitative methodology** involves measurement and analysis of variables between methods (Denzin and Lincoln 2011). This is used for evaluating similar systems for efficiency and accuracy. The evaluation process is objective and is based on an algorithm. In the context of this work QaComPS selection process was quantitatively evaluated against QMP and AHP selection processes.

- **Qualitative methodology** is the set of activates used for observing the behaviour of a system. This is undertaken by a group of experts in the field. This method of evaluation is subjective as it is based on the group's observations. Use of questionnaires and audio visual aids assist the developer in undertaking this methodology. In the context of this research a group of computational modellers were available for applying this methodology.

  Scenarios create real world models of the problem and are used for undertaking qualitative methodology. A scenario describes the problem as a story which describes all the actors (people) involved along with their specific roles and relations (Rosson 2002). A scenario was created for capturing/analysing user's practices, requirements and research problems.

1.3. Research Questions

The study for the novel QaComPS service is driven by the following research questions:

(1) *Which QoS parameters to track and how to effectively update the QoS information?*

A number of QoS parameters are associated with a Cloud provider and choosing the most important parameters and manage those remains to be addressed.

(2) *Can a service ontology effectively describe QoS information and offer a single vocabulary for describing different Cloud providers?*

At the moment there is no standardised way of describing a Cloud provider. Describing the providers in a standard way would simplify the process of comparing the providers.

(3) *How to effectively communicate, semantic information to non semantic services and non semantic information to semantic services?*

The proposed solution has semantic storage of the information while the selection and run are non-semantic. SAWSDL annotations are proposed for offering a seamless channel of communication between the semantic and non-semantic services.

(4) *Does a combination of ranking and selection algorithms perform better than a single selection algorithm?*

Traditionally a single selection is used for the selection purpose. This research proposes a multi stage selection process with ranking and a selection steps. However whether the selection method improves the efficiency and accuracy remains to be answered.

1.4. Thesis Output and Contributions

The output of the thesis is a novel QaComPS service. The QaComPS architecture includes a service ontology for describing Cloud providers and QoS metrics for describing the performance of a provider.

This research has made a number of original contributions. The main contributions of the research include a novel QaComPS service; a cost model capable of translating physical cost (what one pays) into the QoS cost (value for money); and the use of SAWSDL to seamlessly transfer QoS information between semantic and non semantic services. Some of the contributions are as follows:

(1) In the current scenario users are at the risk of getting locked onto a single provider as each provider has its own vocabulary, access protocol and pricing. The semantic descriptions of Cloud providers mean that QaComPS has a single vocabulary for describing providers. It also enables the users to access multiple providers through a single logon.

(2) Quality awareness means that the QaComPS tracks the QoS for each provider thus enabling the user's to make informed selection decisions.

(3) QaComPS has an easy to use interface with a minimalistic number of inputs. This is used for querying the semantically stored Cloud provider's information.

(4) QaComPS effectively and efficiently processes user's queries for Cloud resources by filtering, ranking and selecting the best provider's VM; out of the many available VMs.

(5) QaComPS has the potential to be applied to other domains where quality aware selection is required. Examples include other research domains such where quality aware decision making is required. The design of the QaComPS is loosely dependent on the computational modelling domain.

1.5. Thesis Outline

The thesis is organized into seven chapters; the structure for the remainder of the thesis is given below:

Chapter 2 analyse the Cloud computing against the existing HPC computing solutions. This chapter also presents a review of the different technologies used for developing the proposed solution. These include web services, selection algorithms used for selecting web services and the semantic Web.

Chapter 3 presents an analysis of the current practices. These are analysed in light of the literature. This also includes interviews with the experts in the computational modelling domain. The purpose of the analysis is to identify the research requirements for the envisioned solution.

Chapter 4 proposes the Quality aware Computational Cloud Selection (QaComPS) service, to meet the requirements outlined in chapter 3. The architecture for the QaComPS service was developed as the main deliverable for this research. This chapter describes the five stages involved in the processing of a user’s query. Furthermore the chapter also presents the SAWSDL annotations; these form the communication channel between the semantic and non semantic services.

Chapter 5 presents the implementation of the QaComPS. This includes the individual implementation of three sub services namely Information Management, Selection and Run which form the QaComPS.

Chapter 6 presents the quantitative and the qualitative evaluation of the QaComPS. The quantitative evaluation measured the performance and accuracy of the QaComPS selection process against the QMP and AHP selection processes. The qualitative evaluation was carried out by two experts who were part of the requirements analysis process.

Chapter 7 concludes the thesis by summarizing the major outcomes of this research. The chapter also points to the different directions in which this research can be continued.

Chapter 2
Background to Concepts and Technologies

This chapter provides the background and related work associated for this research. The chapter includes literature regarding Web services in section 2.1, as the main deliverable of this research is a Web service. The selection of Web services is described in the section 2.2. Section 2.3 describes the service ontology and presents research associated with ontology based selection. Service broker and its role in the service oriented computing are described in the section 2.4. Section 2.5 describes the semantic Web along with semantic Web services (SWS). The sixth section describes the Cloud service providers. The next section presents an in depth analysis of the Cloud computing technology against the existing Cluster (in house) and Grid computing technologies. The final section describes the implications of the literature review.

2.1. Web Service

Today a user has a choice of multiple platforms such as Windows, Mac and Linux. These operate differently and software developed for one cannot be used with the other. This lack of interpretability points to a need for software that can be used globally without being tied down to a specific platform. This platform independence is achieved by Web Services which interact over the Internet; using a Web browser (Srivastava and Koehler 2003).

A Web service is a software application that can publish its functions and messages to the rest of the world through the Internet and is accessible through many computing devices. The key advantage of a Web service over a traditional software application is its global accessibility and platform independence (W3Schools 2012).

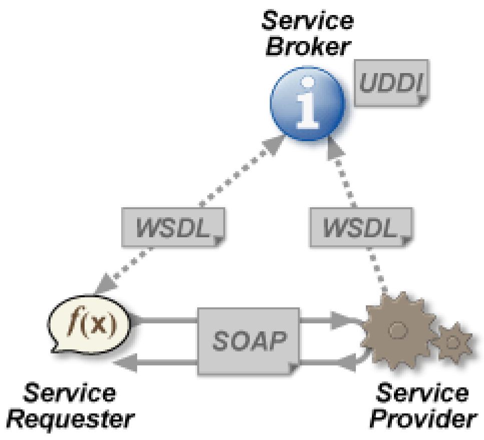


Figure 2.1: Web Service Architecture (Perrey and Lycett 2003)

As shown in figure 2.1 a Web service has three associated parties namely the service provider, the service requester (also referred to as user or consumer) and the service broker. The communication channel between a service user and a provider is Simple Object Access Protocol (SOAP). SOAP uses eXtensible Markup Language (XML) for messaging. The XML message is formatted using Hypertext Transfer Protocol (HTTP) and uses Simple Message Transfer Protocol (SMTP) for transmission (Box, Ehnebuske et al. 2000; Curbera, Duftler et al. 2002).

Web service are described using Web Service Description Language (WSDL) (Curbera, Duftler et al. 2002). WSDL descriptions contain information relating to types, operations and binding. Types describe the type of data being processed such as string or number. The operations is the list of functions that the service can perform, The binding contains details of the physical network necessary for communication for example IPs addresses and ports.

2.1.1. Functional properties of a Web service

Functional properties describe the information associated with the functionality of a service. This consists of service inputs and outputs along with pre/post conditions, associated with the functioning of the service. The information is available through the service provider as it enables the user to select the service. (Ran 2003)

2.1.2. QoS properties of a Web Service

According to Software Engineering Institute (SEI) QoS is “The probability that a system will deliver particular levels of measureable computational and communication properties such as availability, bandwidth, latency, and jitter. Policies and mechanisms typically are designed to control and improve the quality of service of a system” (Linda Northrop 2006). A simpler definition for QoS is given by (Zhou and Niemela 2006) which describes it as a measure of non functional requirements such as reliability and security.

QoS properties are used for evaluating the degree to which a service meets the specified quality requirements. There are two types of QoS properties these are technical QoS properties and managerial QoS properties which are further divided into sub-properties. Technical properties describe the properties related to service operation such as reliability, security, and availability. Managerial properties are associated with the service management such as cost, payment, contract and ownership. (Zhou, Niemela et al. 2007)

A set of sub-properties may be associated with a QoS property for example performance is a measure of response time and latency. Attributes of QoS properties such as complexity, dynamics and unit are required for measuring QoS. The value of a QoS property is either positive or negative such as higher reliability is good while higher cost is bad therefore reliability has a positive value while cost has a negative value. (Tran, Tsuji et al. 2009)

A large amount of research has been done into QoS associated with Web services. This is mainly concerned with the areas of performance, security and trust. Ran et al. has grouped QoS properties into four groups. These are cost, security, runtime and transaction support. Other works have focused on service availability, throughput and response time (Menasce 2002). Tran et al. has described QoS elements as part of an ontology this is described in section 2.4.2.

2.1.3. Evolution of Web Services

In the year 2000 Microsoft proposed Web services which employed XML, SOAP, and WSDL. The key motivation behind Web services was e-commerce. (Levitt 2001).

One of the major advances for Web services was the development of composite services. A composite service combines the functionality of multiple services to reach a specific goal. An individual service can do a single task while a composite service could achieve a large complex task. (Claro, Albers et al. 2006). An example of this could be booking a holiday where a customer has to book a flight, taxi, and a hotel. These tasks can either be booked through three individual services or by one composite service. Composite services are dependent on an effective selection algorithm as these have to identify and select the most suitable service for each task. Web service selection is not only crucial to the effective working of composite services but also vital to the selection of individual services by a user. Therefore multiple approaches to the selection of the Web services have been proposed over the past decade.

2.2. Selection of Web Services

The simplest form of service selection involves matching user's request with the functional properties of the available services. The selection process involves using conditional programming such as if-else constructs. These are used to compare two values and reach a decision. (Pratt and Zelkowitz 1984)

The key issue faced by the functional selection is its inability to differentiate among services offering the same functionality. In an effort to improve the selection new algorithms were proposed which employed QoS parameters. (Serhani, Dssouli et al. 2005). Tran et al. (Tran, Tsuji et al. 2009) argues that:

"With a number of Web services having similar functionality, it is necessary to rank those services to select the best Web services for a request. QoS information which

can reflect user's expectation and experience of using a service is often used as the distinguishing factor in a service ranking algorithm."

The use of QoS parameters is also argued by others, including Godse el al. who states that multiple service providers can match a user's functional requirements thus QoS requirements act as the differentiating criterion *(Godse, Bellur et al. 2011).*

As argued above an effective way of using functional selection is in conjunction with QoS based selection. A number of existing selection algorithms such as Euclidean Distance (Danielsson 1980) and Analytical Hierarchy Process (AHP) (Saaty 2003) have been used for processing the user's QoS requirements.

2.2.1. Euclidean Distance Algorithm

Euclidean distance algorithm is a type of Multi Criteria Decision Analysis (MCDA) algorithm. MCDA is the study of complex decisions with conflicting inputs such as comparing reliability to cost. MCDA decision making involves assigning each criterion a weight and evaluating them explicitly. (Koksalan, Karwan et al. 1984; Triantaphyllou 2000).

Euclidean distance algorithm (Danielsson 1980) is used for QoS based service matchmaking. The matchmaking process is based on the requester's preferences which are compared to the QoS of available services. The service with the smallest distance to the preferences will be ranked highest or in other words; the smaller the Euclidean distance the better the service. The algorithm has been used extensively; in the field of computing (Montanari 1968).

There are three sets of inputs to the algorithm. These are historic QoS data, individual value of each parameter and the relative weight of each parameter. The parameter value represents the individual value associated with each QoS parameter for example 3 out of nine for cost. The parameter weight represents the priority of each parameter compared to the others. For example cost is twice as important as reliability. Historic QoS data builds overtime and reflects the QoS associated with each service on offer. For example provider A has a rating of 7/10 for reliability. The historic data is managed by an independent third party such as a service broker.

The user inputs the required QoS values and the weights. A normalized weight of each QoS parameter is calculated using pair wise comparisons with the user's inputs. The criteria weights indicate the relative importance of each QoS parameter. These are used with the historic data to produce the ranked list of services.(Koivunen 2001).

2.2.2. AHP (Analytical Hierarchy Process)

AHP is also a type of MCDA algorithm, proposed by Saaty et al (Saaty 1980). Unlike Euclidean distance AHP returns a single result.

The AHP based solution has three phases: problem decomposition, comparative judgments, and priority synthesis. The problem decomposition consists of distributing the problem into more comprehendible sub problems as shown in figure 2.2. The comparative judgements are made by pair-wise comparison of each criterion. This used for specifying the relative importance of each criterion. Sensitivity analysis is the final phase in which each alternative solution is combined with the relative local rank to generate the overall ranking. (Saaty 2008)

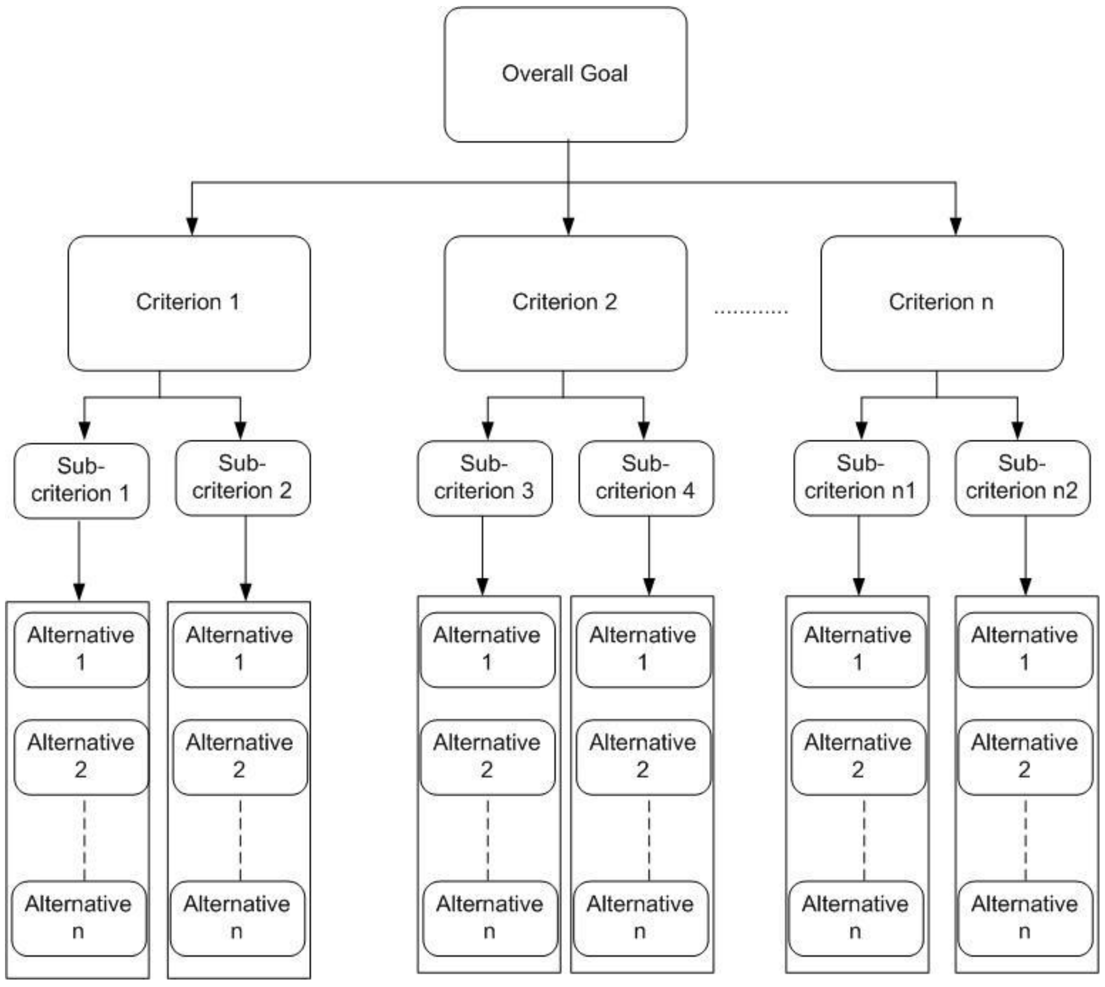


Figure 2.2: AHP hierarchy

Figure 2.2 shows the hierarchy of AHP which is distributed into four tiers. The first tier describes the overall goal which contains a description of the problem along with the objectives. The next tier describes the criterion which affects the decision process. There is no specific number defined as to how many criterions should be used as one can use any number but care should be taken as more criterion means more processing time. The third tier describes the sub criterion; this is optional. The fourth and final section describes the alternatives. Alternatives contain all the possible solution such as when buying a selecting a provider all the available providers would be presented as alternatives. (Saaty 2008)

The AHP algorithm is useful for addressing specific problems and not so useful towards general problem. This is due to the unique criteria and priority associated for each problem. For example an AHP setup for flight booking service cannot be used for hotel booking as both involve a different set of criteria and alternatives.

2.2.3. Quality Matchmaking Process (QMP)

QMP was proposed by Eleyan et al. during his PhD at the University of Manchester. Eleyan et al. describes quality matchmaking as “a process that requires the quality matchmaker to match the quality inquiry to all the quality advertisements”. The matchmaking is performed by the Euclidean Distance algorithm. This measures the nearest Web service to the specifications of the requester.

Inputs to the system are the requester’s quality preferences that are fed to the AHP method. This outputs the quality criteria weights, which are inputted to the Euclidean Distance method. The Euclidean Distance method measures the distance between the user’s quality requirements and the quality specifications specified by the service providers.(ELEYAN, Amna et al. 2004)

QMP is interesting in the sense that it uses Euclidean distance algorithm instead of AHP as the main selection algorithm; as AHP returns a single result while Euclidean distance returns a ranked list. Another aspect of QMP is it’s reduced set of requirements as the weights are calculated. The QMP was upgraded to include a filter, and an interface matchmaking component (ELEYAN 2011).

2.3. Service Ontology

A number of QoS based ontologies for service selection have been proposed these include (Maximilien and Singh 2004), (Zeng, Benatallah et al. 2004), (Tran, Tsuji et al. 2009) and (Godse, Bellur et al. 2011). It is argued by Maximilien et al. that the current Web service standards lack means for expressing a service’s QoS attributes (Maximilien and Singh 2004). The work proposes using ontology for describing QoS attributes; it also proposes a new QoS ontology.

2.3.1 Ontology

Ontology is a formal representation of knowledge as a set of concepts within a domain and their association with one another. These are used for the effective sharing of knowledge and its reuse (Gruber 2008). A domain is a set of entities which share a common interest, for example football, cricket and hockey fall under the domain of sport. Domains can be connected with other domains and can also have sub domains. This enables domain experts to design their ontology and connect it to larger parent ontologies. There is a wide array of online ontologies; produced by W3C (W3C 2012) and many others (Smith and Grenon 2002; Jaiswal, Avraham et al. 2005; Ruebenacker, Moraru et al. 2007; Cassidy 2008).

2.3.1.1. Ontology Components

In order to ensure interoperability between ontologies a common structure is followed irrespective of the language used for expressing the ontology. Common components associated with ontologies are described below (Gruber, 2008)

(1) **Individuals:** represents the instances and objects associated with ontology. For example University is a concept while University of Leeds is an individual as it physically exists.

(2) **Classes:** concepts in Ontology can be presented as Classes and Subclasses.

(3) **Properties:** the object properties describe the connections between classes while the data properties describe the data associated with the classes.

(4) **Relations:** represent how classes are related to one another such as child class, parent class.

(5) **Restrictions:** associated with properties are used for verifying inputs.

(6) Events: mark the changing of attributes or relations.

2.3.2. WS-QoSOnto

Tran et al proposes a QoS ontology namely WS-QoSOnto and associated QoS based ranking algorithm for selecting Web services (Tran, Tsuji et al. 2009). The proposed solution consists of a service ontology for describing QoS properties, and relations. It is proposed that WS-QoSOnto will improve the process of Web service selection which is a key prerequisite to an effective implementation of composite Web services. The proposed selection process uses AHP selection algorithm.

Tran et al. has evaluated WS-QoSOnto against a variety of existing QoS models including DAML-QoS (Zhou, Chia et al. 2004), WSMO-QoS (Li and Zhou 2009) and OWL-Q (Kritikos and Plexousakis 2007). The ontology presented as part of this work is very descriptive and QoS parameters specified in that ontology are part of this research. WS-QoS ontology facilitates specification of QoS at different quality levels while the AHP algorithm implementation has fared well for selection.

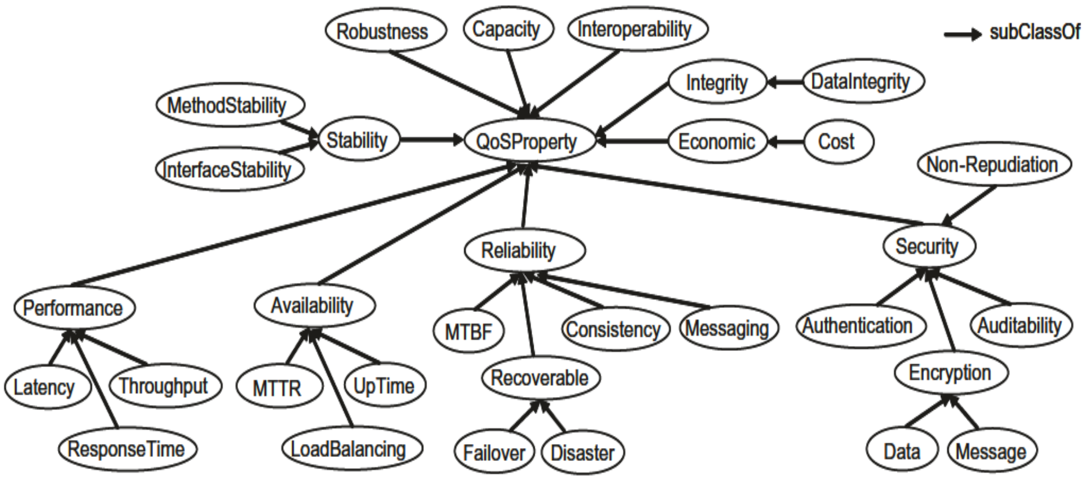


Figure 2.3: Core QoS properties(Tran, Tsuji et al. 2009)

This work also includes a list of core QoS properties as shown in figure 2.3. There are ten high level properties including reliability, security and economic. There is also a range of sub-properties such as cost which is a sub-property of the economic. The list of properties shown is not exhaustive but does contain the most commonly addressed properties. Out of these cost, security and reliability are extensively studied as described in section 2.1.3. These three parameters are also part of this research and are further described in chapter 4.

2.3.3. Other research

An alternative service ontology is given in (Godse, Bellur et al. 2011). This research also features reliability, reputation and security QoS properties among others. The research has identified major service selection elements and their categories. These are performance, correctness, security, reputation while reliability was a sub-criterion of performance.

A service ontology is proposed by (Wang, Sun et al. 2010) which considers correctness of the elements such as robustness and accuracy over the lifetime of each element. The need to add trust and reputation for the selection of Web services is presented in (Wang and Vassileva 1007). This work reports on their comprehensive investigation of trust and reputation systems in other areas. It provides valuable observations and approaches that can be used in Web service systems. This work also includes a typology to classify Web services from three aspects, centralised vs. decentralised, persons/agents vs. resources, global vs. personalised.   The first aspect is of interest since the broker is an example of decentralised system.

2.4. Service Brokers

The role of a traditional service broker is to produce and monitor SLAs (Service Level Agreements). Service SLA contains the formal definition of the service and is written by the service provider. Often these are legal binding between providers and users and are managed separately for each service transaction. However this role varies depending on the provider. For example, service providers which permit negotiation can be negotiated with by the broker.

The service broker can act at three levels; (a) an Information broker which can only offer information; (b) an information broker with decision making which can not only share information but using selection algorithms identify the best information; (c) a broker with information, selection and the ability to acquire the service and set it up for the user.

A service broker uses functional properties for identifying the matching services while most also use QoS properties in the process (Zhou and Niemela 2006). The broker is independent of provider's influence and can make an independent assessment of the QoS ratings. to collect and store QoS (Altintas, Berkley et al. 2004). The general approach for the broker is to update the QoS database after every transaction; thus building up a historic record. For example Lin et al (Fairley 2007) propose the broker should collect user ratings after each transaction in order to build up the reputation database for all the services. By delegating trust management to brokers, individual users only need to ask their brokers about the reputation of a service before any transaction with a server. In addition, brokers can form a trust network where they exchange and collect reputation data about services. The only overhead for a user is the responsibility to share the reputation the feedback with its broker. Finally Serhani et al (Serhani, R. Dssouli et al. 2005) present a two-phase verification technique for Web services. The first phase consists of syntactic and semantic verification of the service interface description including the QoS parameter's description. The second phase consists of applying a measurement technique to compute the QoS metrics stated in the service interface and compares their values with the claimed one. A similar approach is used in (T.Rajendran, Dr.P.Balasubramanie et al. 2010) their WS-QoS broker architecture.

The service brokers are extensively researched by the Grid community. The most relevant service brokers to this research are AssessGrid (Djemame, Padgett et al. 2011). The AssessGrid had a QoS based SLA broker with risk assessment support. The risk assessment component evaluated the probability of a SLA failure. Along with the risk assessment the broker has provision for negotiation for resources within the Grid infrastructure. Service QoS has been investigated by Djemame et al. in the context of Cloud computing (Armstrong and Djemame 2009).

2.5. Semantic Web Services (SWS)

This sub-section first provides an overall description of the semantic Web followed by details of the two main types of semantic web service (SWS).

2.5.1. Overview of Semantic Web

The current Web is for humans only, and consists of billions of Web pages which are linked to one another. The semantic Web aims at converting the current Web of documents to a Web of data. The semantic Web is for both humans and machines as it provides meaning to the content of the Web pages and describes relationships between entities. The semantic Web provides a common framework that allows data to be shared and reused across application, enterprise and community boundaries (Miller 2001).

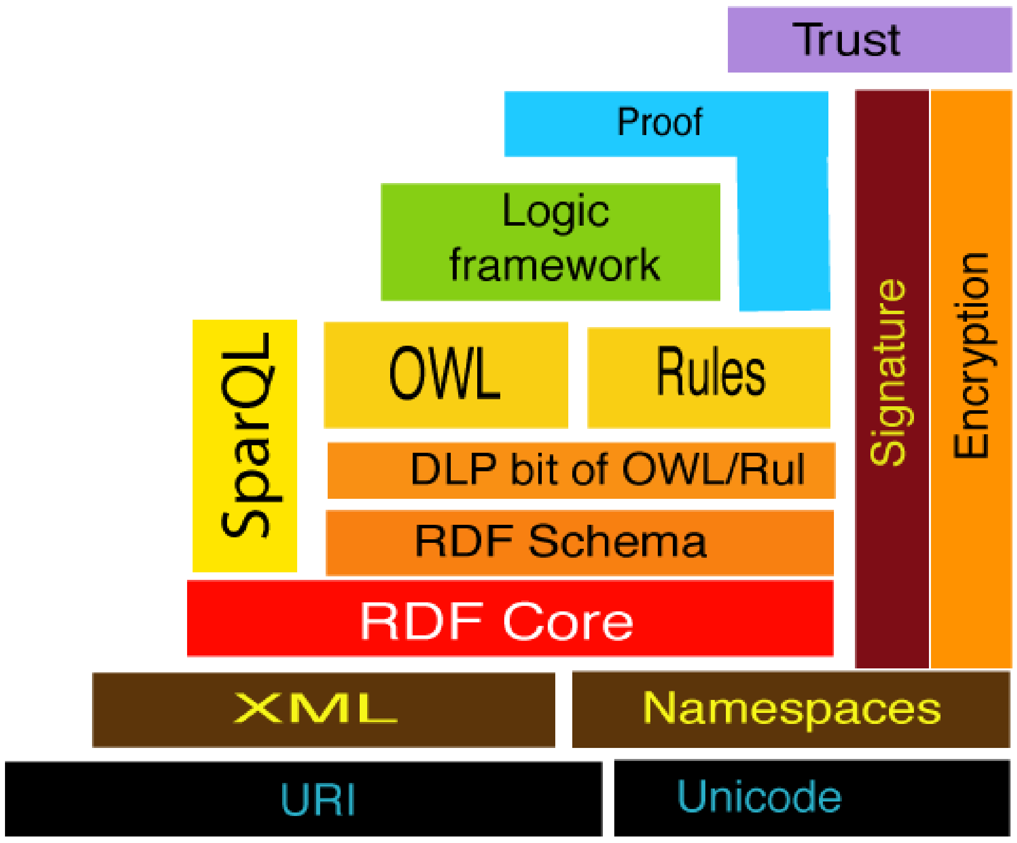


Figure 2.4: Semantic Web stack(Bratt 2007)

The concept of a semantic Web was put forward by Sir Tim Berners Lee (Berners-Lee and Hendler 2001). He is founder of the World Wide Web (WWW) and the director of World Wide Web Consortium (W3C). W3C leads the collaboration of researchers and industrial partners which is working towards realizing the semantic Web (Herman 2008). It is mainly concerned with two things: common formats for grouping data which is extracted from a range of varied sources; and language for describing how the data is connected to the real world objects.

The semantic Web stack as shown in figure 2.4 represents the semantic Web architecture. The stack is a hierarchical representation of language and technologies that are necessary for realizing the semantic Web. Each layer of the stack exploits and uses capabilities of the layers below.

2.5.1.1. Uniform Resource Identifier (URI) and Unicode

URI is a global naming scheme used to identify content on the Web. They offer high level of flexibility through sub-types such as URL (Uniform Resource Locator) and URN (Uniform Resource Name). The URL is used for identifying Web resources such as Web pages for example http://google.co.uk or http://leeds.ac.uk. (Masinter, Berners-Lee et al. 2005). URN is used for identifying objects for example ISBN numbers (Balani 2005). URI are global and can be created and owned by anyone. These offer an effective means for naming resources and objects on the Web.

Unicode is the computing industry standard for encoding, representing and processing text from most of the world's major languages. The current version of Unicode consists of 110,000 characters. (Consortium 2000)

2.5.1.2. Extensible Markup Language (XML) and Namespace

XML is a markup language that is used for encoding documents in a format that is interpretable by both humans and machines. XML is defined by XML specification which is managed by the W3C (Bray, Paoli et al. 1997). There are a number of languages which are described using XML specification these include WSDL and Resource Description Framework (RDF) (Berners-Lee, Handler et al. 2006).

XML namespaces are identified by a URI and are used for introducing uniquely named elements and attributes in an XML document (Bray, Hollander et al. 1999).

2.5.1.3. RDF (Resource Description Framework)

RDF provides a consistent, standardized way to describe and query Internet resources from text pages, graphics, audio files and video clips. It also supports syntactic interoperability among different semantic services (Balani 2005).

RDF is a W3C specification for representing semantic information on the Web. It is ideal for representing metadata about Web resources. RDF information is processed by software applications rather than humans. It provides a common platform for exchanging of information between applications. (Manola, Miller et al. 2004)

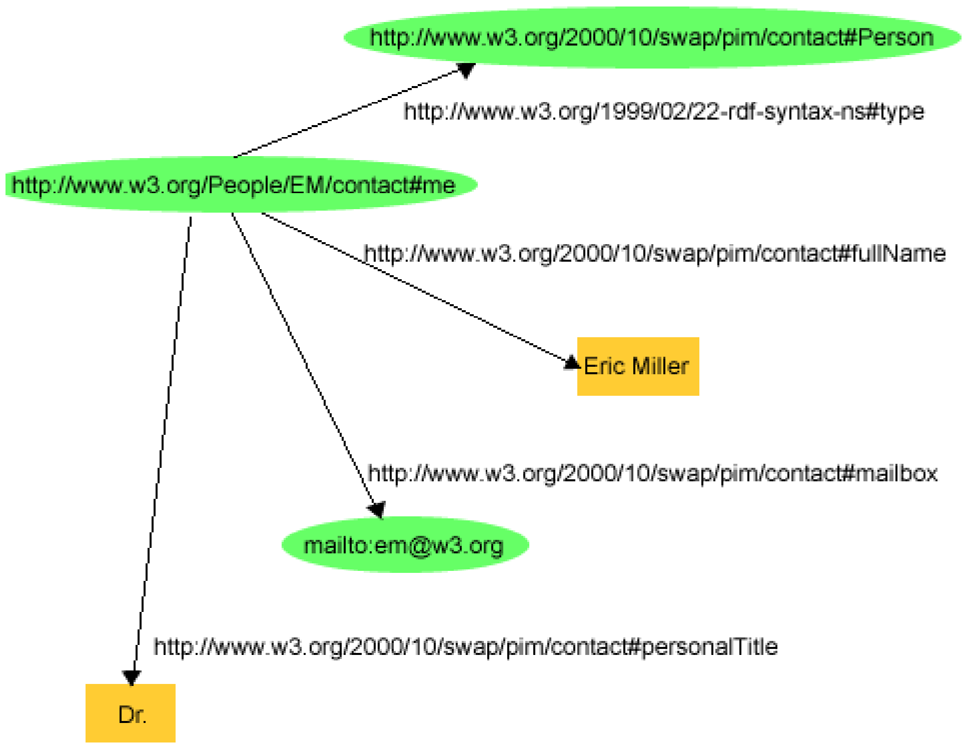

http://www.w3.org/2000/10/swap/pim/contact#Person
http://www.w3.org/1999/02/22-rdf-syntax-ns#type
http://www.w3.org/People/EM/contact#me
http://www.w3.org/2000/10/swap/pim/contact#fullName
Eric Miller
http://www.w3.org/2000/10/swap/pim/contact#mailbox
mailto:em@w3.org
http://www.w3.org/2000/10/swap/pim/contact#personalTitle
Dr.

Figure 2.5: RDF graph describing Eric Miller (Manola, Miller et al. 2004)

RDF is based on identifying resources through URI using properties and property values. The property values along with the resources represent RDF graphs.

An RDF graph is a collection of nodes and arcs where a node represent a subject or object while an arc represents a predicate. A subject and an object is an individual like John or a thing like chair. In terms of English grammar a subject or object is a noun or pronoun. A predicate shows the relationship between a subject and an object; grammatically a predicate is a verb describing an action or state. An RDF graph describing Eric miller is shown in figure 2.5. The graph shows four sets of linked information the first being “me is a type-of person”, second being “me personal title is Dr”, third being “me mailbox is em@wm.org” and the fourth being “me full name is Eric Miller”. This graph describes “Dr Eric Miller is a person who can be reached at em@wm.org”. As RDF is semantic, a software application would actually understand this statement and could share it or use it. An RDF graph is written down in the form of RDF triples where each statement in the graph is a triple. The graph in figure 2.5 consist of four RDF triples.

There are many semantic languages associated with RDF. These include Resource Description Framework Schema (RDF-S), and SPARQL Protocol and RDF Query Language (SPARQL).

- **RDF-S**

  RDF-S is a semantic extension of RDF and provides mechanisms for describing groups of related information along with their relationships (Balani 2005). It defines common vocabularies for RDF data models (Pan and Horrocks 2007). These vocabularies describe properties and classes which provide the basic descriptions of ontologies.

- **SPARQL**

  SPARQL is a query language for querying RDF graphs. It was designed specifically to meet the use cases and requirements identified by RDF (Prud'Hommeaux and Seaborne 2008). SPARQL syntax is similar to that of the widely used Structured Query Language (SQL).

  A typical SPARQL query consists of prefix, select, update and where clauses. The prefix clause is used for abbreviating URIs such as

  “PREFIX dc: http://purl.org/dc/elements/1.1/” indicates that dc will be used instead of the given URL. The select clause is used for retrieving data while update clause is used for updating. In the following example entries from the title column are being selected. ‘WHERE’ clause is used for specifying the conditions of selection. In the following example all the entries are being selected from the given URL.

```
PREFIX  dc:  <http://purl.org/dc/elements/1.1/>
SELECT  ?title
WHERE   {?x dc:title ?title}
```

2.5.1.4. Web Ontology Language (OWL)

OWL (McGuinness and Van Harmelen 2004) is an ontology language based on RDF/XML. Compared to RDFS it facilitates greater machine interpretability of Web content. It also provides a wider vocabulary for describing classes and properties. OWL also offers descriptions of relationship between classes, cardinality, equality and a richer type set of properties.

Like RDF-S, OWL can define classes, sub classes, create instances, specify relations using object properties and associate values to instances through data properties. It can also identify that two classes are disjoint, and identify distinct individuals or that a data property is functional or non functional. (Horrocks, Patel-Schneider et al. 2003)

2.5.1.5. Semantic Web Summary

The semantic web was proposed by Sir Tim Berners Lee who by creating HTML played a vital role in the emergence of the current Web. According to (Berners-Lee and Hendler 2001); semantic Web is the next step in the evolution of the current Web.

The semantic Web stack illustration as shown in figure 2.4, was created by Sir Tim Berners Lee. As of now semantic technologies up to OWL (Web Ontology

Language) are standardized these are being used to build semantic Web applications such as semantic Web services.

2.5.2. Introduction to Semantic Web Services (SWS)

Semantic Web services are a synergetic confluence of semantic Web and Web services. They have the potential to offer value-added services by automatically discovering and assembling web services to accomplish a domain task. The overall philosophy for SWS is also referred to as service oriented computing (SOC). (Nandigam, Gudivada et al. 2005)

There are a number of active researchers and W3C working groups are striving to introduce semantics into Web services. Their proposed solutions can be sub divided into ontology based semantic Web services and annotation based semantic Web services.

2.5.2.1. Ontology Based SWS

A Web service ontology is used to provides with a set of semantic markup languages to provide the conceptual model (Bruijn, Bussler et al. 2005). This facilitates fuller automation of Web service tasks, such as Web service discovery, execution, composition and interoperation (Martin, Burstein et al. 2004). However due to its differences with the existing WSDL based service oriented architecture, there are limitations to the implementation of these services. In case of existing Web services converting them to SWS through this approach involve a complete rewrite of the service.

A number of such Web service ontologies have been proposed these include OWL-S and WSMO (Web Service Modelling Ontology) (Vitvar, Kopecký et al. 2008).

OWL-S is an ontology language used for describing semantic Web services. It builds on OWL and proposes to enable users and software agents to discover, invoke, compose, and monitor Web resources with a high degree of automation.

OWL-S ontology has three main parts: the service profile for advertising and discovering services; the process model, which gives a detailed description of a service's operation; and the grounding, which provides details on how to interoperate with a service, via messages. (Martin, Burstein et al. 2004)

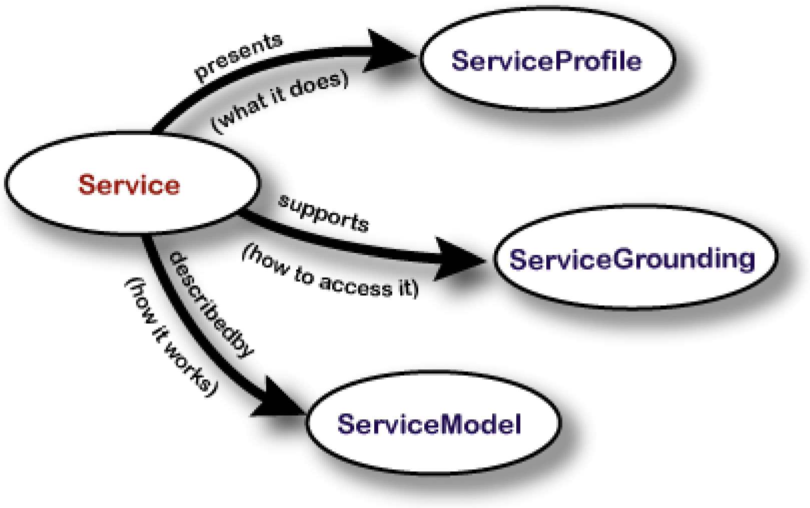


Figure 2.6: OWL-S Ontology (Martin, Burstein et al. 2004)

Figure 2.6 show the three parts of OWL-S ontology where service profile describes the function of the service, input requirements and limitations of the service. The service model describes how to connect to a service by detailing the semantic content of the requests and the responses. Service grounding specifies the details on how to connect to a service. It specifies message formats, communication protocol, and other details for contacting the service (Martin, Paolucci et al. 2007). The biggest challenge facing OWL-S is in regards to service grounding as currently there is no standard execution platform for OWL-S (Lara, Roman et al. 2004).

An alternative to OWL-S is Web Service Modelling Ontology (WSMO). This provides a conceptual framework and a formal language for semantically describing Web services with machine-process-able semantics. It helps to facilitate the automation of discovering, combining and invoking electronic services over the Web (Vitvar, Kopecký et al. 2008).

Recent work by Fensel et al. presents a formal analysis of OWL-S against WSMO and other semantic technologies. This concludes that mediation is a key issue facing the wider realization of semantic Web services. Regardless of the differences WSMO also face issues in regards to service grounding which acts as a hurdle to its adoption.(Fensel, Facca et al. 2011).

2.5.2.2. Annotation-Based SWS

Annotation-based SWS adds semantic annotations to WSDL documents. Web Service Description Language Semantic (WSDL-S) and Semantic Annotations for Web Service Description Language (SAWSDL) are the two annotation based SWS solutions being worked on by W3C groups. (Akkiraju, Farrell et al. 2005; Kopecký, Vitvar et al. 2007)

WSDL-S is a W3C member submission for introducing semantics to Web services. It is as an evolutionary and compatible update to the WSDL. The proposed changes include annotating the capabilities and requirements of the Web service with references to a semantic model. This is achieved by annotating service inputs, outputs and operations. Mechanisms to specify and annotate preconditions for the Web service are also part of the WSDL-S.

The WSDL-S specification was last updated in 2005. In 2006 SAWSDL was proposed which followed the same key design principles of WSDL-S. It replaced WSDL-S by introduced many new concepts.

SAWSDL adds semantic annotations to WSDL documents which point to semantic concepts for specifying semantics or schema mappings for data transformations. It shares some key principles with WSDL-S these include:

- Building on the existing WSDL framework and adding semantic annotations
- Using semantic annotations to discover and invoke Web services

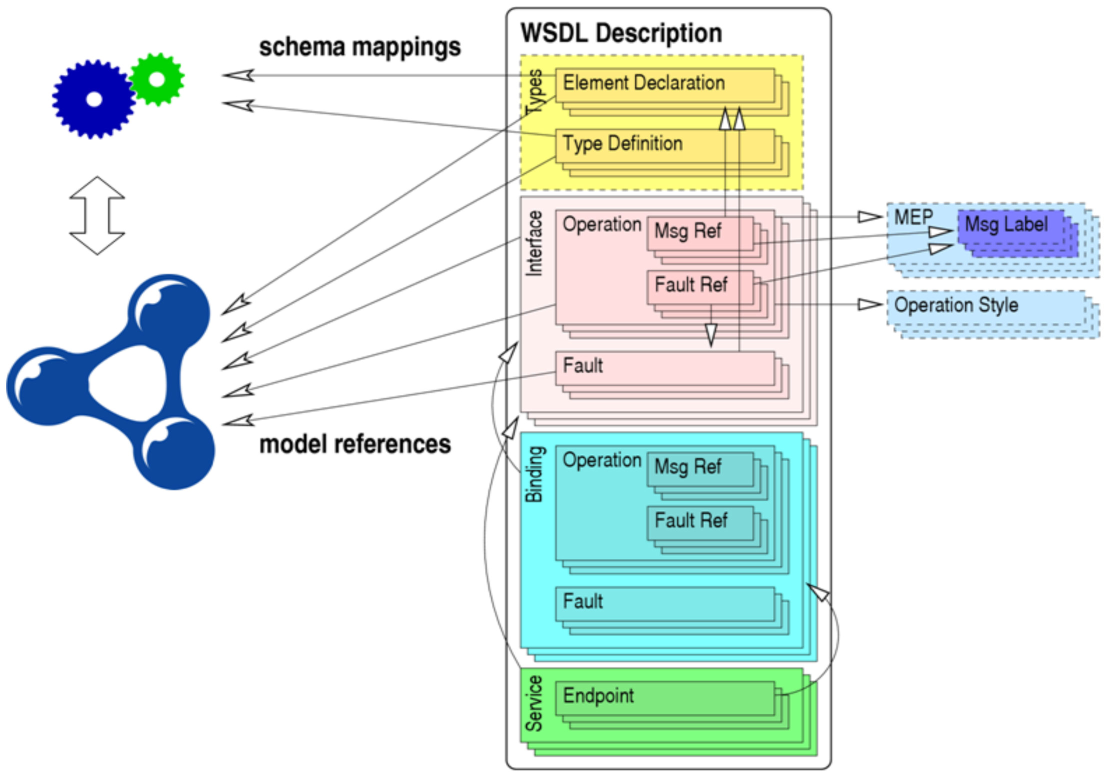


Figure 2.7: SAWSDL (Kopecký, Vitvar et al. 2007)

The notion that SAWSDL is an extension of WSDL is presented by figure 2.7 which shows a SAWSDL document. The block with the title WSDL description represents the WSDL document while the rest are SAWSDL extensions.

A model reference is a set of URIs relating to a concept in semantic model. It is used for providing semantic annotations to WSDL elements. The schema mappings are used for the transfer of information between semantic and non semantic sources. There are two types of schema mappings namely lifting schema mapping and lowering schema mapping. Lifting schema mappings are used for acquiring information from non semantic sources while lower schema mapping is used for sending information to non semantic sources.

2.5.3. Advantages of Semantic Web Services

The key advantage of SWS is the enhanced the level of automation for Web service discovery, composition and invocation. Other advantages include standardization of naming schemes and standard format for the description, storage and exchange of data. (Sheila 2001; Sirin, Hendler et al. 2003; Sycara, Paolucci et al. 2011)

2.5.3.1. Automating Web Services

The introduction of semantic markup to Web services enables them to understand the meaning of a document. Typical Web services can understand the series of characters that make up the words in a document. These however do not understand what these words mean (Balani 2005). Being able to understand the meaning leads to automating the processes associated with a Web service.

(1) Automatic Discovery

Sheila et al. was among the first few people to identify that automatic service discovery was possible by using semantic markup. An example of automatic discovery would be: a user wishing to buy an airline ticket from Leeds to London. In case of current Web services a user would start with a search engine to find the list of Web pages which offer the required service. The user would then read each Web page in order to identify whether it offers the required service. After identifying all the pages that offer the required service, user would make a selection. In this scenario a semantic Web service would be able to locate the appropriate Web services as the description of each service is understood by the search engine (Sheila 2001). The semantic Web service would return only the related results.

(2) Automatic Composition

Camara et al. proposes ITACA which is an integrated toolbox for the automatic composition and adaptation of Web services (Camara, Martin et al. 2009). This states that automatic service composition and service reusability can be achieved through rich service interface description.

ITACA toolkit is used for specification and verification of adaptation contracts, automates the generation of adapter protocols. It is developed using Python (Beazley 2009) and Java (Horton 2011).

2.5.4. Semantic Web Services Conclusion

Semantic web services have been defined as an amalgam of semantic Web and Web services. The semantic Web services introduce meaning to the message exchanges between services thus introducing automation to the process of service discovery and composition. Service composition refers to a group of one or more services working seamlessly together to achieve a complex task. (Nandigam, Gudivada et al. 2005)

The key motivation behind semantic Web services is the same as that of Web services i.e. e-commerce. These services have the added advantage of higher levels of service description, automatic service discovery and composition (Sheila 2001).

2.6 Cloud computing Services

According to National Institute of Science and Technology (NIST) “Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.”(Peter Mell 2011)

The NIST definition of Cloud proposes three types of Cloud services namely Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS) (Peter Mell 2011). These are further described in the following section.

- ***SaaS:*** The provider offers software applications to the user as a service. These are running on the providers Cloud infrastructure and are accessible remotely from various client devices through a thin client interface such as a Web browser. The underlying infrastructure including network, servers, operating systems and storage are not controlled by the user whose rights are limited to the user-specific application. Examples include GoogleApps.

- ***PaaS:*** This service enables a user to build and run a service on the Cloud infrastructure remotely. The Cloud providers offer support for a specific set of development tools such a programming languages which are used by the user for development and deployment. The provider has an underlying Cloud infrastructure which includes processing servers, storage servers, operating systems and network. The ownership of the application is with the user however the control for the underlying architecture is with the provider. Examples of PaaS providers include Microsoft Azure (Microsoft 2012) and Salesforce (Salesforce 2012).

- ***IaaS:*** IaaS service enables users to acquire processing, storage, network and other computing resources remotely. These resources are managed by the provider with the user being able to deploy and run software systems. Examples of IaaS include (Amazon 2012; FlexiScale 2012; GoGrid 2012; Rackspace 2012).

  A Virtual Machine (VM) is the key resource on offer from an IaaS Cloud service. It is an emulation of a physical machine (Computer). The VM is created by deploying a VM Image; i.e. is a configured set of software which includes the operating system. A VM image can be deployed very quickly as it takes a few minutes in most cases to setup a VM. The physical machines are controlled by the Cloud provider while the VM are in control of the user.

The VMs are accessed in many ways including online interfaces and SSH clients. (Buyya, Yeo et al. 2009)

2.6.1. Cloud Service Deployment Models

There are four deployment models for Cloud services; these are private, community, public, and hybrid. A deployment model indicates the attributes associated with Cloud services specially the access attributes. (Peter Mell 2011)

- ***Private Cloud Services:*** The Cloud infrastructure is used and operated by the same organization. This is a highly trusted and secure model as in most cases the infrastructure is based locally within the organization. The disadvantages of this model include lack of elasticity; i.e. increasing or decreasing the size of the Cloud on-demand.

- ***Community Cloud Services:*** The Cloud infrastructure is shared and operated by a group of organizations, with all supporting policy, security and operations.

- ***Public Cloud Services:*** The Cloud infrastructure is available to the general public or business for use. This is owned by a large organization and is the most common form of Cloud deployment. Large organizations such as Amazon, Microsoft and Google offer this form of Cloud.

- ***Hybrid Cloud Services:*** The Cloud infrastructure is a combination of two or more types of Clouds. This model requires the sub models to be bound by standard set of communication rules. An example of this would be a community Cloud working with a public Cloud to handle untimely surge in resource demand.

2.6.2. Cloud Computing Advantages and Challenges

This section describes advantages of Cloud, and the challenges that are hindering the growth of Cloud computing. A detailed study of the advantages and challenges is given in (Armbrust, Fox et al. 2010).

2.6.2.1. Advantages of Cloud computing

The following advantages are unique to Cloud computing as neither Gird nor Cluster computing offer these benefits.

- On-demand Access (access to an any amount of resources at any time of the day)
- Eliminate upfront costs
- Ability to pay per use
- Economies of scale due to very large data centres
- Higher utilization by multiplexing of workloads
- Simplify operation and increase utilization via resource virtualization
- Replication (Running multiple copies of the same job for ensuring reliability)

Advantages such as on-demand access and replication save user time and effort. In case of Grid or Cluster every user’s job is submitted to a queue while in case of Cloud the user can access any amount of resources at any time. Replication increases the reliability of the service as due to this feature a user’s job will not be interrupted at times of disaster. Eliminating upfront costs, pay per use, high utilization are of interest to supervisors or people in-charge as these help with managing the costs. (Armbrust, Fox et al. 2010)

2.6.2.2. Challenges faced by Cloud computing

A list of challenges faced by Cloud computing is identified by (Armbrust, Fox et al. 2010) and is given in Table 2.1. The list is compiled from the user's perspective as it points to issues such as getting locked down to a single provider or losing resource availability.

Index	**Challenge**	**Solution**
1	Ensure availability of resources	Use multiple Clouds
2	Data Lock-In	Introduce mechanisms to access multiple Cloud through single account
3	Data confidentiality	Monitor security, Deploy encryption
4	Building trust	Mechanisms for accessing providers past performance
5	Performance unpredictability	Manage the provenance data of the Cloud, Improve VM support
6	Scalable Storage	Mechanisms to scale the load among the different storage options
7	Bugs in large distributed systems	Introduce debugger
8	Scaling quickly	Invent Auto-Scaling tool
9	Data bottle necks	Improve bandwidth
10	Software licensing	Pay-per-use licences

Table 2.1: Key challenges and their solutions (Armbrust, Fox et al. 2010)

Challenges like data management, security, privacy, service provisioning and Cloud economics are identified by (Dikaiakos, Katsaros et al. 2009). The study states that Cloud data is stored at a number of un-trusted hosts which introduces security and privacy loop holes. Another security and privacy challenge is identified in (Arshad, Townend et al. 2009). It states that in IaaS Cloud multiple VMs run on a single physical machine which can transpire into security and privacy threats as all the VM share the physical machines memory.

2.6.3. HPC Grids and Computational Clouds

IaaS Cloud computing services are a type of computational service. Computational services offer physical resources such storage, communication and processing in the form of services. These services have remained in extensive use by the scientific research community and to some extent by the industry. (Foster, Kesselman et al. 2001). Cluster computing and Grid computing are the antecedents to Cloud computing.

2.6.3.1. Cluster Computing

A computer cluster is a collection of two or more computers used for undertaking compute intensive problems. A cluster consists of a set of tightly coupled computer systems with a centralized job management and scheduling system. All the computers in the cluster use a single system image thus the whole cluster behaves as a single entity.

Computer clusters were in use even before the emergence of Web services. However they have evolved over the years and now are used remotely through a network. The advantage of such a system is the sheer amount of compute power on offer. (Bader and JáJá 1999)

2.6.3.2. Grid Computing Services

Grid computing is focused on large scale sharing of computational resources such as storage and processing (Foster, Kesselman et al. 2001). The concept of Grid computing was proposed by (Foster, Kesselman et al. 2001). It proposed the creation of a computational Grid for solving large compute intensive tasks.

Grid computing is build on research in the field of Cluster computing as a computational Grid is a set of loosely coupled computing machines or clusters. The Grid is a hardware and software infrastructure that provides dependable, consistent pervasive and in-expensive access to high-end computational capabilities. The grids were created by pooling computational resources from a number of organizations to serve a common purpose. The pooling required hardware infrastructure to achieve the necessary interconnections and software to monitor and control the resulting ensemble. (Foster and Kesselman 2001)

Grids were very popular among research institutes specially universities and a number of computational grids were created these include the White Rose Grid (Dew, Schmidt et al. 2003), Nordu Grid (Eerola, Kónya et al. 2003), and the Sun Grid (Gentzsch 2001). Issues faced by computational grids include large upfront costs, associated with buying hardware and software resources. A more pressing issue is regarding the resource allocation as due to the large number of users, every user’s job are submitted to a job queue.

In recent years the usage of Grids has been reduced due to the availability of Cloud. One such example is National Grid Service (NGS) offering Cloud services. NGS is the largest public sector provider of HPC resources in the UK. It serves a number of educational and research organizations (NGS 2012).

2.6.3.3. Semantic Grid

The semantic grid is an extension of the computational grid where information and services are given well defined meaning, enabling people and machines to work in cooperation (De Roure, Jennings et al. 2005). The semantic grid initiative was part of the UK e-science program. The key requirements for the semantic grid are as follows:

- Resource Description, Discovery, and Usability
- Process Description and Enactment
- Autonomic Behavior
- Security and Trust
- Annotation

Semantic resource descriptions are effective for describing QoS properties. These properties are have been extensively used by Grid based resource brokers and are also used in this research to describe Cloud providers resources (Serhani, Dssouli et al. 2005).

The semantic descriptions lead to autonomic behavior, this includes automatic service discovery, dynamic service function and automatic service maintenance. As described in section 2.5.3 one of the key advantages of introducing semantics is automation. The proposed solution involves dynamic behavior and automatic QoS maintenance.

Semantic annotations are effective for describing the QoS properties of Grid and Cloud resource providers. The annotation based approach was employed by Semantic Annotations for Web Service Description Language (SAWSDL). This research uses SAWSDL for communicating between semantic and non semantic services.

Research towards semantic grid has dwindled during the past five years. This may be attributed to the advent of Cloud computing.

2.7. Cost analysis of Cloud computing against HPC

A number of studies have been conducted to compare the economics of Cloud computing against the HPC setups. These include an economic analysis by (Armbrust, Fox et al. 2010), a cost-benefit analysis by (Kondo, Javadi et al. 2009), a performance analysis by (Ostermann, Iosup et al. 2010) and a performance analysis for scientific computing by (Iosup, Ostermann et al. 2011).

Early comparative studies of HPC Cloud providers are given in (Geelan January 22, 2009) and (Evangelinos and Hill 2008). Another paper (Deelman, Singh et al. 2008) investigated the value of using EC2 HPC option compared with "in-house" HPC provision, for three large e-science problems. These are Montage (multi scaled images); Broad (earthquake USGS); and epigenetic (heritable changes in gene expression). Each application involved a pipeline workflow passing a file between each stage of the workflow. This work reported that it's very easy to move to a HPC Cloud like EC2 but there is a trade-off between memory and computation resources. And that the time to start and stop the job needs to be carefully planned. The time management is due to the difference levels of performance by the Cloud at different times of the day this is due to the variation in workload. Further they found TeraGrid (an in-house HPC solution) was more expensive (using their cost model) and pointed out the latency in launching compute jobs led to much larger turn-round times for comparable resources.

There are a number of emerging academic Cloud provides such as UK's National Grid Service (NGS), (Chang, Wills et al. 2011) and Open Cloud Consortium (OCC) (Consortium 2009). Pricewise Clouds are reasonably cheap for scientific computing as the cost of hiring a VM is very low. Although the VM is cheap additional costs such as bandwidth can augment the total cost. (Iosup, Ostermann et al. 2011).

Seti@home is a scientific experiment that uses Internet-connected computers in the search for Extraterrestrial Intelligence (SETI) (Anderson, Cobb et al. 2002).

Running SETI on Amazon EC2 for a year would cost 7000 USD which is 60 percent of the current cost of running SETI for a year. The bandwidth costs in this case make more than half of the overall cost. (Kondo, Javadi et al. 2009).

Some studies argue that the cost of acquiring Cloud resources for very large periods of time (over a year) is much more expensive compared to the HPC. (Kondo, Javadi et al. 2009; Ostermann, Iosup et al. 2010; Iosup, Ostermann et al. 2011).

The main shortcoming of these studies has been acquiring resources for months and years on an hourly basis. Cloud providers do offer monthly and yearly options which are much cheaper than the hourly option. For instance Amazon charges $0.90 per hour for a small EC2 machine; however if the same machine was to be bought for a period of three years it could cost as low as $0.2 per hour. (Amazon 2012).

It is also argued that the costing in most studies favour the HPC setup as extra costs such as space, power, and insurance are not considered. There are two reasons for this, firstly these amenities are free as the parent institute bears the expanses and secondly it is very difficult to calculate costs such as air-conditioning or UPS (Hazelhurst 2008).

2.7.1. Performance analysis of Cloud computing against HPC

Three areas of performance are presented in this section. These are compute performance, performance benchmarks, and performance variability or variance. The compute performance is measured by running an experiment on the Cloud and the HPC and comparing the completion times. Benchmarks are standard points of reference used for comparison. A Cloud based VM is benchmarked against a physical machine for comparing different aspects of performance such as maximum load or data transfer. The performance variability or variance measures the inconsistency among different sets of results for an experiment. This is achieved by running the experiment several times on the same machine.

2.7.1.1. Compute performance analysis

Amazon EC2 Cloud has been compared against two cluster based HPC setups by (Hazelhurst 2008). The comparison was made by running a scientific application.

	Total number of nodes	**Total number of processing cores**	**Total amount of memory (GB)**
Amazon EC2	32	64	240
C4	37	74	148
iQudu	160	320	2560

Table 2.2: Specifications of HPC clusters

The specification of the two HPC options and Amazon EC2 are given in Table 2.2. It is worth noting that all three options had different amounts of computing resources with Amazon having the smallest number of processing cores and one tenth the memory of iQudu.

The results shown in figure 2.8 indicate that Amazon EC2 cluster acquits itself well. As with just 32 nodes it achieves an efficiency value of 72% while iQudu with 160 nodes maintains 90% efficiency.

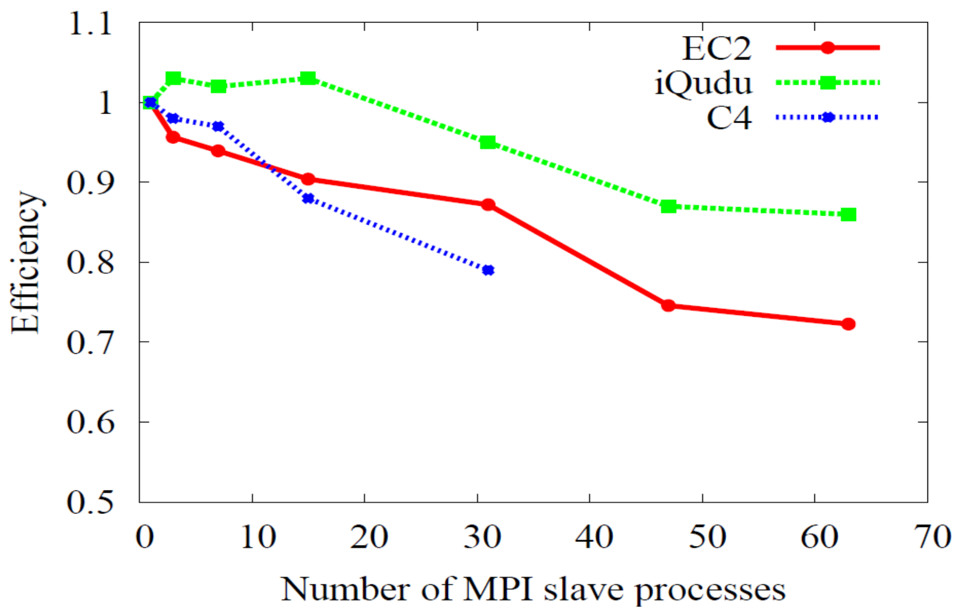


Figure 2.8: Cost effectiveness of Cloud versus HPC (Hazelhurst 2008)

2.7.1.2. Cloud performance benchmarks

NASA Advanced Supercomputing (NAS) benchmarks are used for benchmarking the performance of HPC options. These include Embarrassingly Parallel (EP), Message Gateway (MG), Block Tri-diagonal (BT), Conjugate Gradient (CG) and Fourier Transform (FT) benchmark (Bailey, Barszcz et al. 1991).

- EP: Provides an estimate of highest achievable limits of a machine.
- MG: It is used for testing both short and long distance data communication.
- CG: Used for measuring irregular long distance communication and matrix vector manipulation.
- FT: A 3-D partial differential equation solution using Fourier Transform on many special codes for long distance communication performance.
- BT: It solves synthetic system of partial differential equations. It also serves as an input/output benchmark.

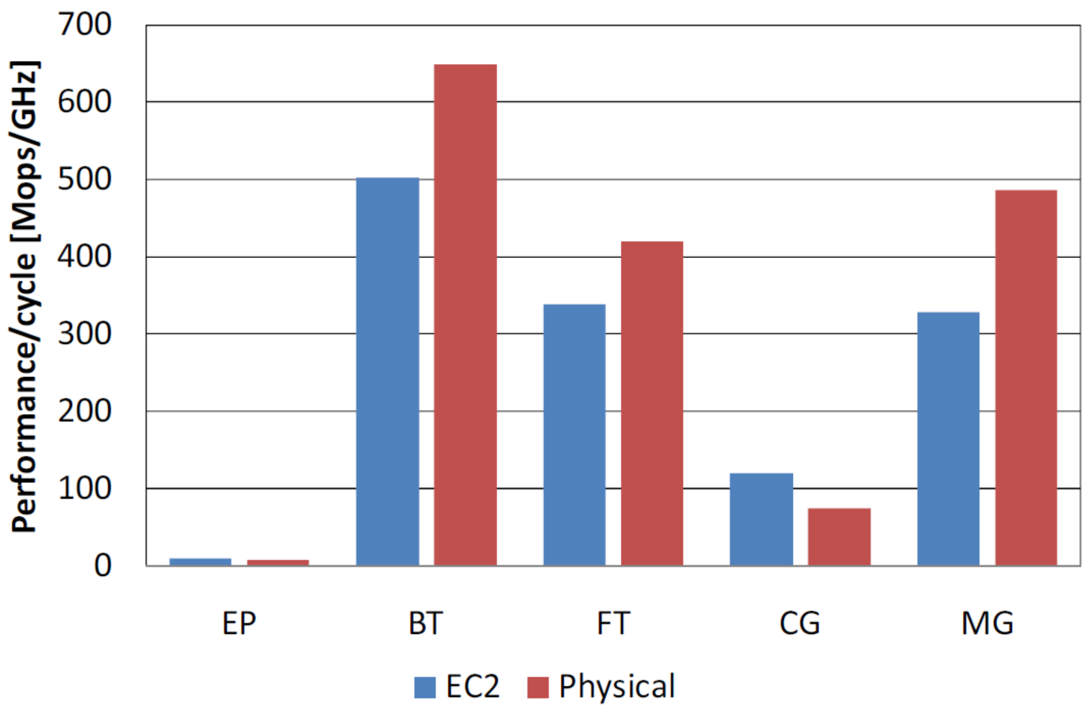


Figure 2.9: Performance comparison between an Amazon EC2 VM and a physical machine (Higher is better) (Akioka and Muraoka 2010)

Amazon EC2 VM and a similar physical machine were benchmarked by (Akioka and Muraoka 2010) as shown in figure 2.9. Compared to the physical machine the VM performed better for EP and CG. The VM was outperformed for BT, FT and MG. The VMs performance for BT is attributed to the inappropriately small size of the BT benchmark. In case of MG and FT the cause of low performance was bandwidth and network latency. (Hazelhurst 2008; Akioka and Muraoka 2010)

2.7.1.3. Cloud performance variability

Cloud services are usually affected by performance variance as shown in figure 2.10. Figure 2.10a shows the performance variance for running jobs on Amazon EC2. The different runs show a compute variance between 5% and 10%. In case of data transfer the variance is far higher than computing as shown in figure 2.10b.

The main cause of variance in Cloud is the varying amounts of workload on the Cloud. This is magnified by the on-demand nature of the Cloud which means that the provider can potentially handle an infinite amount of user requests. (Kondo, Javadi et al. 2009; Jackson, Ramakrishnan et al. 2010; Iosup, Ostermann et al. 2011)

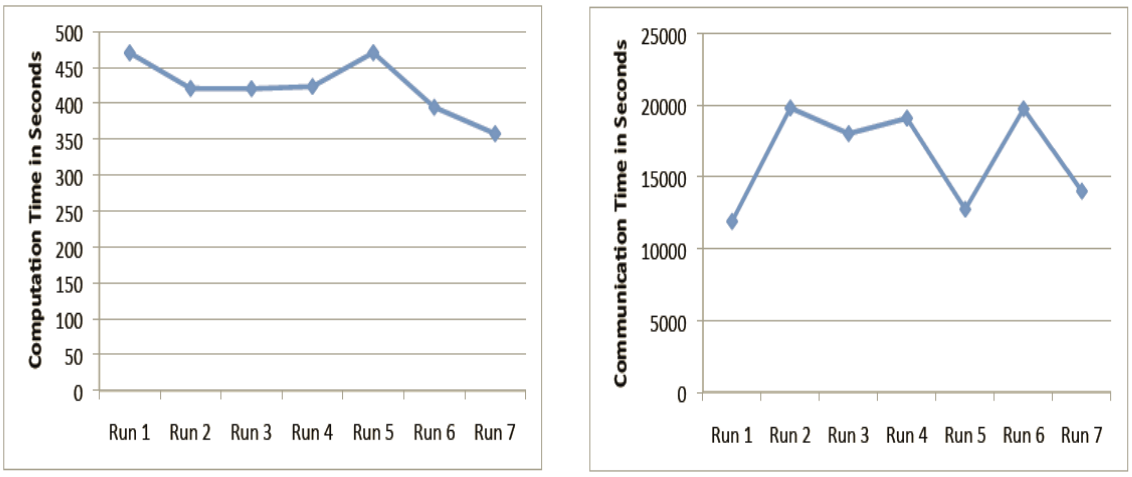


Figure 2.10: Performance variance for Amazon EC2 (Jackson, Ramakrishnan et al. 2010)

In case of HPC setups the variance is less significant as every job is submitted to a queue and a fixed number of jobs can run at any time. Even in times of surge in demand; the wait associated with running the job would be longer but efficiency would remain the same.

2.7.2. Cloud security and usability:

Clouds are operated through data centres with each having as many as five thousand physical computers (Greenberg, Hamilton et al. 2008). The software and hardware configurations and support platforms within a data centre are homogenous. This homogeneity means that each machine will have the same security setup that would be managed centrally at the data centre. Access to the Cloud is one of the security challenges as it involves user names and passwords which could be compromised. Another aspect of security is that the Cloud is always remote and access is through a third party Internet Service Provider (ISP). (Foster, Yong et al. 2008)

HPC setups are built over a period of time; as due to the funding constraints most institutions procure the resources step by step. This means the HPC setup tends to be a locally managed heterogeneous system. However this does not affect the level of security as access to these resources is strictly limited. Only a group of authorized users working at the same organization can access the HPC setup. Some of the technical aspects to the security of HPC and Cloud are given below. (Nelson, Dinolt et al. 2011)

- Authentication: This identifies the user and his access rights on the Cloud or HPC setup. Each Cloud provider has its own unique login mechanism such as Amazon; which has a two level login mechanism dependent on security certificates, usernames and passwords. In case of HPC setup users are assigned security credentials at the start of their tenure.
- Storage: In case of Cloud multiple copies of the data are stored at geographically distinct locations. These copies of the data on separate geographic locations do improve the reliability however they do pose a challenge for ensuring security of data. Considering the amounts of data being transferred encryption and decryption of data alone is a major challenge. In case of HPC setups data is stored locally and has fewer security threats.

2.8. Implications of Literature Review

Web services are a very effective way of communication as these are both globally accessible and platform independent. Semantic Web Services offer all the functionality of a traditional Web service with the added benefits of higher levels of service automation and better description of data. This research employs Web services for communication with the users. SAWSDL annotations are used for sharing the semantically stored RDF information.

The thesis addresses the needs of computational modellers (users) for running large HPC jobs. Computational modellers at the moment use either Cluster or Grid computing for accessing HPC resources; required for solving large scientific problems. This thesis proposes the use of IaaS Cloud computing. Advantages of Cloud computing include on-demand access, replication, no upfront costs and high utilization. Cloud computing is beneficial to the users as it offers a reliable and efficient way of running their jobs. It also attracts the public and research organizations by a reduction in costs and high reliability. A number of Cloud service providers offer different services these are described using service ontology while the data is stored as RDF.

The efficient and effective selection of Cloud services is an important part of this research. The proposed solution uses Euclidean distance for ranking and AHP for selection; in a novel method for selection. It uses the two algorithms together to reach the best (cheapest) Cloud provider’s service.

Chapter 3
Analysis of Current Practice

Computational modellers require large amounts of computing resources for undertaking HPC experiments. The current practice for acquiring these resources is to use the local HPC setup which is usually a collection of computing clusters. An alternative to the current practice of using HPC is Cloud computing. Cloud has come to the attention of computational modellers as Cloud services are suitable for undertaking HPC jobs. The move towards Cloud has been swift as up until three years ago the definition of Cloud was being contended (Geelan 2009) while today, users have a variety of Cloud services. These include public Cloud services (available to everyone) such as Amazon EC2, educational Cloud services (available to research community) such as NGS Cloud service, and private Cloud services (available to members of a specific organization) such as IBM’s Blue Cloud (Wang, Tao et al. 2008; Li, Yang et al. 2010; Sultan 2010).

This chapter describes the process of requirement analysis for identifying the research requirements. These are used in the later chapters for architecting the envisaged system and building its prototype. The evaluation process is also dependent on these requirements. The steps involved in the process of requirement analysis are shown in figure 3.1.

The chapter is sub divided into four sections where the first section gives an overview of the research problem. The next section describes the current practice for acquiring computational resources. The third section describes an Information broker. The broker was developed as part of the initial experiments to develop an understanding of the working brokers. It also helps with gathering the technical requirements for building the prototype of the envisaged system. The final section describes the requirements for the envisioned system.

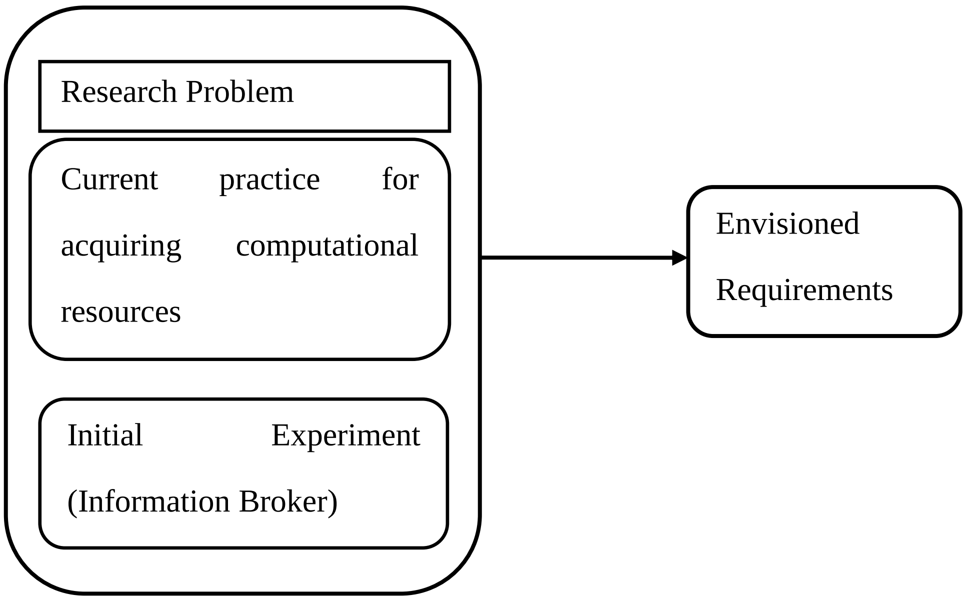

Research Problem
Current practice for acquiring computational resources
Initial Experiment (Information Broker)
Envisioned Requirements

Figure 3.1: The requirement analysis framework

3.1. Overview of the research problem

Cloud providers offer on-demand access to a vast amount of computing resources. This makes Cloud ideal for addressing computational modeller's needs for running HPC jobs. However there are issues such as Cloud provider's "best-efforts" policy. In a "best-efforts" where the provider can just drop the service and provides no guarantees concerning the response time, throughput etc. For example see the Amazon EC2 Service Level Agreement (SLA) (Amazon 2012). In case of Cloud computing, users have to make decisions based on trust and reputation rather than guarantees. In such a scenario information regarding providers past progress would be very useful in making the selection decisions. Quality of Service (QoS) data presents a historic record of the provider's progress with parameters for identifying rates of success and failure. (Foster, Roy et al. 2000)

Cloud computing providers offer computing resources as Virtual Machines (VM). The VMs are classified by size of resources the offer such as Small, Medium, Large and Very Large. Cloud providers typically offer four VM sizes with some offering more options. As the VMs on offer share the functional properties it is the

QoS properties that act as the service differentiating criteria (Godse, Bellur et al. 2011).

The case study for this research state that; a computational modeller's requirements will vary during the modelling life cycle. For example during the early stage of model development a cheap, less reliable Cloud service is often sufficient as failed jobs can be easily repeated. As the model development matures much larger computational recourses are required; now the reliability (number of jobs that succeed) and the reputation of the provider become more important.

The research problem is to provide a service, that can mediate across a number of computational Cloud providers, to select and run (transparently) the best (e.g. cheapest) computational provider's VM, that meets the user's computational and QoS requirements.

3.2. Current practice for acquiring computational resources

The study of the current practice has been conducted in collaboration with experts from the field of computational modelling. The modellers have access to a local HPC setup for running compute intensive jobs. They include a Professor (hereon referred to as Professor A) and a doctorate student (hereon referred to as Student B) working in the modelling group in University of Leeds.

3.2.1. Current practice: Introduction

The local HPC setup at the University of Leeds consisted of computing clusters and very large computing servers. The last chapter reported on the field of cluster computing (see section 2.6.2.1). People involved in the current practice include: computational modellers, laboratory managers and lab directors. Inputs from these experts are vital in pursuing this research; as they would outline the requirements and expectations for the envisioned system. A brief description about the people and their roles is given below.

- **Computational Modeller**

  Position: Student A is a PhD student in a scientific computation group.

  Research Interests: Developing efficient, accurate and reliable computational techniques for the solution of partial differential equations. The majority of his research work has involved the *Elasto Hydrodynamic Lubrication (EHL) problems*.

  Priorities: Deriving results of interest to the wider scientific computing community, for publication in papers and thesis. Develop modelling framework for EHL problems.

- **Laboratory Manager**

  Position: Laboratory Manager C is a research assistant who is responsible for managing access to the local HPC setup.

  Research Interests: C is interested in supporting researchers performing HPC simulations of fluid flow for EHL.

  Priorities: Managing the service which entails managing the job queue and recovering the service in case of a failure.

- **Laboratory Director**

  Position: Professor B is the laboratory director for the scientific computing group.

  Research Interests: Professor B’s general research area is Scientific Computing specifically he is interested in efficient, accurate and reliable computational techniques for the solution of partial differential equations. The majority of his research work has involved with the development and analysis of computational algorithms using HPC systems.

Priorities: Acquiring funding for maintaining the local HPC setup. Responsibilities also include keeping supervising the service managers.

3.2.2. Interviews with the experts

Semi-formal interviews and discussions were conducted with the computational modeller and the lab director. A selection of questions and answers from these interviews are as follows.

3.2.2.1. Computational modeller interview

The interview with the computational modeller was held on the 26th of August 2009 at the computing lab.

Question 1: Describe your need for HPC resources.

Answer: "The research experiments are performed iteratively where each step involves generating a computational mesh and using it to run the experiment. The code for generating the mesh needs only minor changes for each run. The code for the actual experiment is written individually per iteration. Over the lifecycle; the size of the mesh varies between some hundred Mega Bytes (MB) to tens of Giga Bytes (GB). The local HPC resources are used for running the experiment while meshes can be generated on the local machine."

Question 2: Describe the process of acquiring HPC resources.

Answer: "The process starts by lodging a request for required resources. This is submitted to a job queue. The request includes information on the functional requirements and the time for which the resources are required. The functional requirements include memory, processing and storage these outline these identify the required resources. Requests that require more than twelve hours of time are catered only during the weekends."

Question 3: Describe the advantages of using the current HPC setup.

Answer: "The research experiments would take days on a personal computer while on a HPC machine these take hours. Therefore availability of large amounts of HPC resources is the main advantage."

Question 4: Describe the disadvantages of the current HPC setup.

Answer: "The key issues faced by me in undertaking this work include waiting for job execution as each job is submitted to the job queue. The wait is worse for jobs longer than 12 hours as these are catered only during the weekends. The other issue is, in regards to specifying the amount of time it takes the job to finish. It is very difficult to predict the precise time a job takes to finish and one can only predict probable times. If the prediction is incorrect and the job takes longer, it gets dropped and one has to start from scratch. Another issue is due to the fixed resource size; which means that there are no provisions for coping with surges in resource demand. At times of high demand the queues get longer which translates into more waiting time for each job.

3.2.2.2. Lab Director Interview

The interview with the computational modeller was held on the 3rd of September 2009 at his office.

Question 1: What are the benefits of the current HPC setup?

Answer: "The key benefit is in regards to the higher levels of trust and security. As the physical resources are located and managed locally and access is only granted to authorized members, from within the research group."

Question 2: What are the key issues affecting the current HPC setup?

Answer: “The main issue is in regards to cost; this high upfront costs of buying the local HPC resources and maintenance costs for managing the resources. Another concern is redundancy as HPC resources become obsolete within a span of five years”

Question 3: What are your expectations from an envisaged system?

Answer: “There are a number of expectations from the envisioned system. These include:

- Reduction in costs
- Multiple options for resource sizes
- Instant access to computational resources
- Ability to cope with surge in resource demand
- High levels of security and reliability ”

3.2.3. Current Practice Summary

In the current practice the computational modeller used HPC resources from the local HPC setup. In the current setup the modeller’s progress was being hampered due to the delays in running long jobs as these were entertained only during the weekends. The modeller was also frustrated with the job request process which involved specifying the amount time for which the resources were required. As the time slots were booked in advance, jobs overrunning their time would be dropped which means rerunning the job. The lab director was most concerned about the reduction in costs; and the higher requirements for reliability and security.

3.3. Problem scenario for the evaluation

Ben is a 2nd year PhD student working in the Computational Modelling Laboratory at the University X. He is researching new mathematical models for simulating the fluid flow in a chemical reactor. The model code is developed iteratively. The iterations typically involve code development, running the code, and analysing the model output against the results from a physical experiment. If the model results are not acceptable the code is changed and re-run. After completion of the initial model development stage Ben needs to run much larger models that require much larger amounts of resources.

Typically Ben requires modest computational resources which are addressed by a Cloud provider. On average Ben requires 4-6 one hour time slots each day. He needs modest computing resources and is less concerned about the reliability and the provider’s reputation of the service. This means that cost is the key priority and lower cost computational services can be procured.

As Ben’s model development proceeds to create new science Ben needs larger computational resources with higher levels of reliability and good user’s reputation ratings. The cost of the resources is still quite important to Ben.

Just prior to a demonstration of their latest computational modelling results to their industrial financers; Ben needs very large computational resources with a higher level of reliability and a Cloud provider with good reviews from the previous user’s (reputation ratings). The laboratory is happy to pay a higher rate for 2-3 one-off runs having got a quote for the resources required.

Details of Ben’s computational requirements are set out in Tables 3.1. The QoS requirements are given in Table 3.2. The ratings are subjective and are scored on the nine point scale with 1 being the lowest.

Turnaround time (hour)	Computing Memory (GB)	Storage (GB)	Processor (Cores)	Download (GB)	Upload (GB)
1	2	50	1	10	10
12	4	250	4	15	15
8	8	250	8	25	25

Table 3.1: Ben’s functional requirements

Cloud Provider’s Cost	Cloud Provider’s Reliability	Cloud Provider’s Reputation	Security
9	2	2	2
6	8	5	6
3	9	9	9

Table 3.2: Ben’s QoS requirements

3.4. Information broker for selecting the best Cloud provider’s service

Information brokers are impartial sources of information; addressing user’s requirements for information regarding a specific field. In the context of this research Information broker was identified as a first step towards addressing the research problem. The Information broker was developed in collaboration with Klacnik et al. and the research findings were published in (**Tomaž Klančnik** 2009). An initial version of the broker was developed by the author of this thesis. The design of the final version of the broker was developed in collaboration with Klacnik et al who was also responsible for the development and the evaluation of the final version. The envisioned Information broker would offer users impartial and valuable information regarding functional and QoS properties of Cloud providers.

The broker: helps users, formulate queries; identify information sources that are relevant to a query, and process the queries to identify the best solution to users queries. A first step to building an Information broker was to define a “similarity” measure between the provider's data and the user's requirements. Similarity is a quantity that reflects the strength of relationship between two objects or two features. However in many cases measuring the dissimilarity (i.e. distance) is easier. This measure can then be normalized and convert it to the similarity measure (Teknomo 2006).

The use of Euclidean distance search algorithm for manipulating QoS parameters is given in (Taher, Basha et al. 2005). This research also presents a QoS data model for storing information and a QoS Manager for processing information. The Information broker used the Euclidean distance algorithm for identifying the similarity/dissimilarity between the requests and the resources. The Information broker also had a QoS Manager component for processing user's queries. This QoS component operates in a novel way as compared to the aforementioned QoS Manager.

In addition to similarity measurement, QoS ratings for reliability and reputation were identified for measuring the trust worthiness of the providers. This was achieved by adding a reputation rating system where every user rates the provider after observing the outcome of their transaction. Binary rating (i.e., success vs. failure) is known to be adequate for calculating a reputation value. The reputation value is representative of the expected outcome of a transaction with a specific provider (Papaioannou and Stamoulis 2006). The reputation system used for the experiment is described in (Jsang and Ismail 2002). The system is based on the beta probability density function which is used to represent probability distributions of binary events.

3.4.1. Design of the Information Broker

The Information broker addresses user's requirements for Cloud resources. It had Computational Manager, Resource Manager and Reputation Updater for managing provider's functional and QoS information. The QoS Manager component was for processing user's queries as shown in figure 3.2.

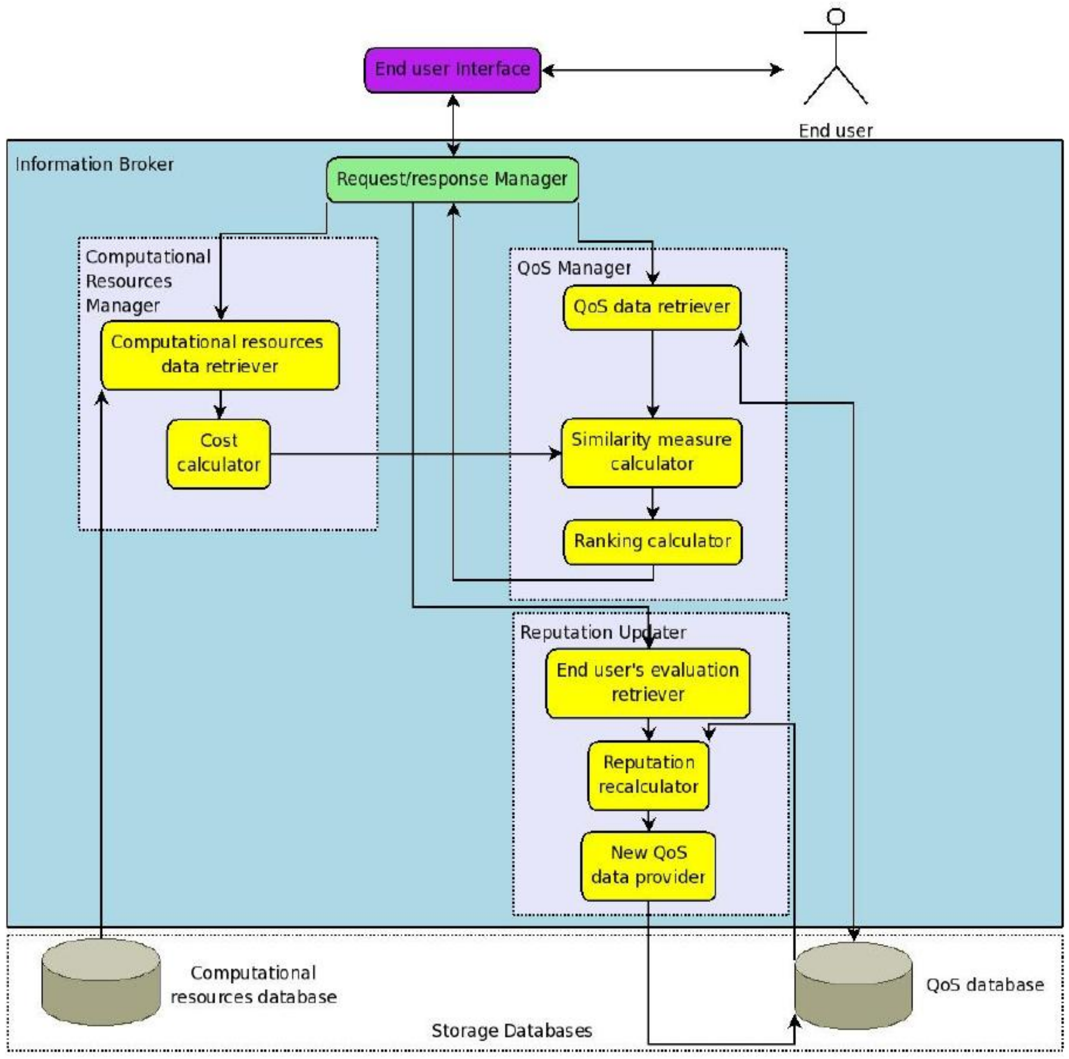


Figure 3.2: Information Broker Architecture

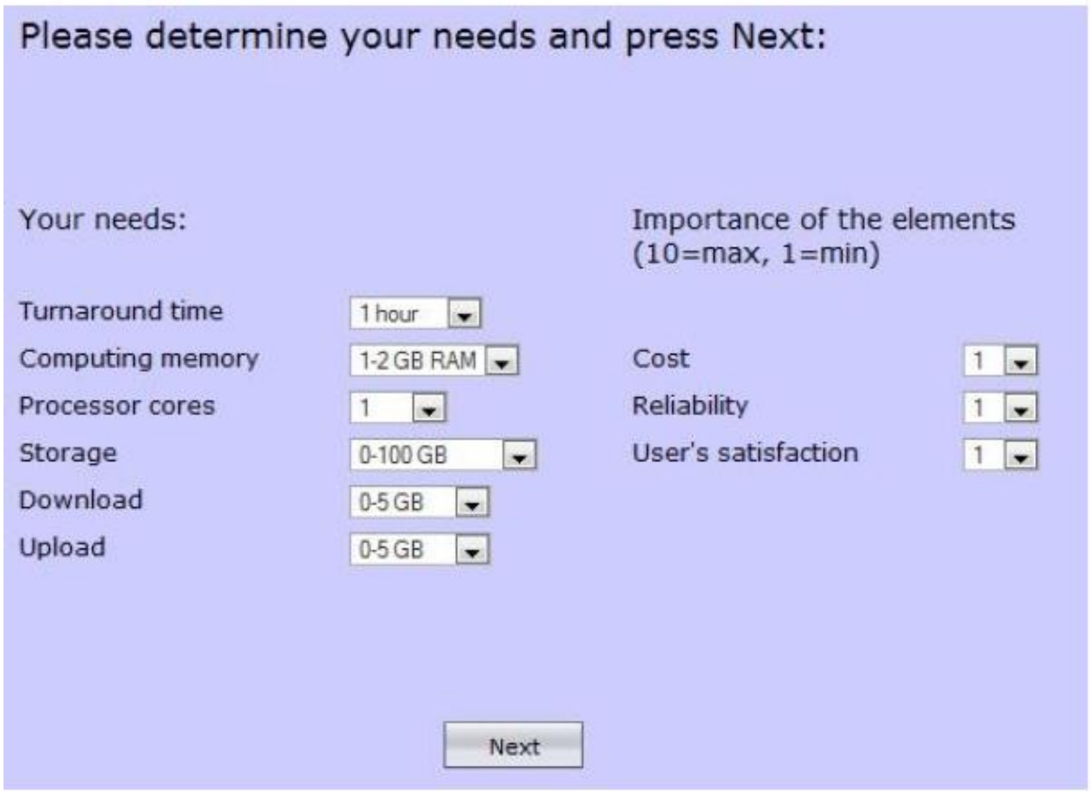
The Computational Resources Manager was responsible for calculating cost of user's requested resources. It retrieved data from broker's computational resources database and from the user requirements. After getting the data the broker calculates the cost of user's requested computational resources for each Cloud provider.

The QoS Manager is the central component of the Information broker. It contains three elements: QoS data retriever, Similarity measure calculator and Ranking calculator. The reputation factor is split into reliability and reputation which are based on previous user's evaluations. The Similarity measure is used for calculating the similarity between user's requirements and the provider's resources. Similarity is measured by Euclidean distance algorithm; which returns a list of ranked providers.

Reputation Updater's task is to collect user's evaluation data. After each transaction the user is requested to provide his evaluation of the selected provider's resources. This is used to update the reputation rating of the provider in the QoS database. The beta reputation system was used for calculating reputation.

3.4.2. Implementation of the Information Broker

The user interface for the Information broker is shown in figure 3.3. The interface enables the user to provide inputs for functional requirements (on the right side) and QoS requirements (on the left side). The functional requirements include amount of physical resources such as memory, processing and the turnaround time which indicates the time requirement. The QoS requirements are inputted on a scale of one to ten with ten being the highest.



Please determine your needs and press Next:

Your needs:

Turnaround time	1 hour
Computing memory	1-2 GB RAM
Processor cores	1
Storage	0-100 GB
Download	0-5 GB
Upload	0-5 GB

Importance of the elements (10=max, 1=min)

Cost	1
Reliability	1
User's satisfaction	1

Figure 3.3: Information Broker User Interface

3.4.3. Formative evaluation of the Information broker

The Information broker was evaluated by the problem scenario. This describes the practice for acquiring computational resources. The aim of the evaluation was to identify: (a) the information management and processing ability of the Information broker; (b) the search capability of the Information broker; and (c) requirements for envisioned system.

3.4.3.1. Evaluation experiment setup

The evaluation experiment was carried out by introducing three simulated Cloud providers. The functional aspects of these providers were based on actual Cloud providers such as Amazon EC2. The functional data regarding provider services is shown in Table 3.3 while the QoS data is shown in Table 3.4.

Provider	**Cost**			**Memory**	**Processor**	**Storage**
	VM cost	**Bandwidth Upload Cost**	**Bandwidth Download cost**			
Prov1vSmall	0.0675	0.06	0.11	2	1	160
Prov1Small	0.135	0.06	0.11	4	4	320
Prov1Medium	0.270	0.06	0.11	8	8	650
Prov1Large	0.540	0.06	0.11	15	8	1650
Prov1xLarge	0.540	0.06	0.11	8	20	1650
Prov2vSmall	0.053	0.10	0.10	2	1	100
Prov2Small	0.105	0.10	0.10	4	2	200
Prov2Medium	0.210	0.10	0.10	8	4	400
Prov2Large	0.372	0.10	0.10	16	4	800
Prov2xLarge	1.348	0.10	0.10	16	8	1600
Prov3vSmall	0.135	0.07	0.10	2	1	100

Prov3Small	0.27	0.07	0.10	4	1	200
Prov3Medium	0.440	0.07	0.10	6	2	400
Prov3Large	0.680	0.07	0.10	8	4	800
Prov3xLarge	1.360	0.07	0.10	8	8	1600

Table 3.3: Provider's functional values

It can be seen in Table 3.3 that same VM sizes have offer different amount of resources. For example large VM for Prov1 has 15GB memory with 8 processing cores while the same for Prov2 has 16GB of memory with 4 cores. These differences are in line with the actual providers, as these do not have a standard definition for the resource sizes. For instance Amazon EC2 small machine has 1.7 GB of memory while FlexiScale small has 1 GB memory (Amazon 2012; FlexiScale 2012).

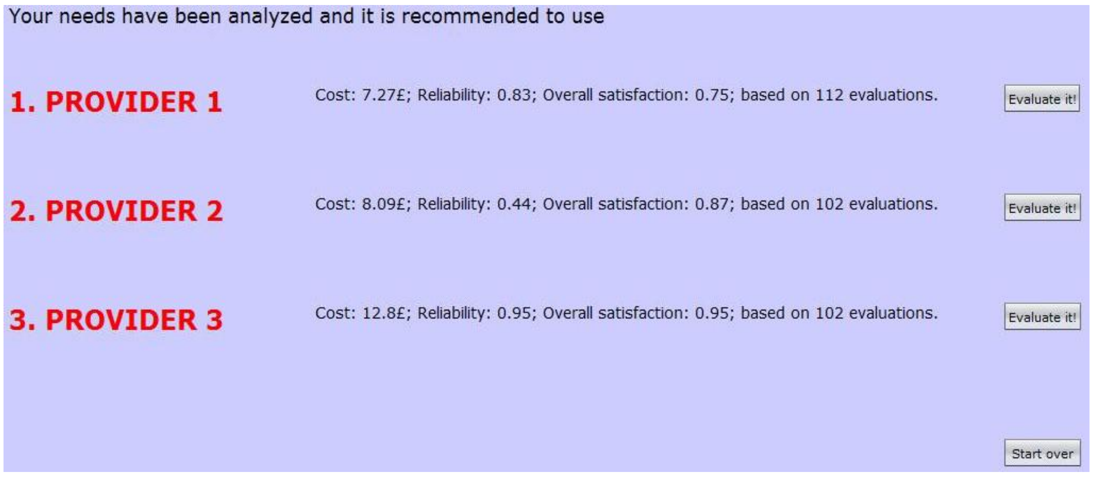
Provider	**Reliability**	**Reputation**	**Number of user evaluations**
Prov1	0.83	0.75	112
Prov2	0.44	0.87	102
Prov3	0.95	0.95	102

Table 3.4: Provider's QoS values

The QoS values were collected at the provider's level instead at the VM level. The decision was made to reduce the processing times. The Information broker did not cater the QoS requirement for security as there were no means to measuring security.

3.4.4. Evaluation results

The experiment was carried out on a machine with a 2.5 GHz Intel core 2 duo processor, 4GB memory and 250GB of storage.

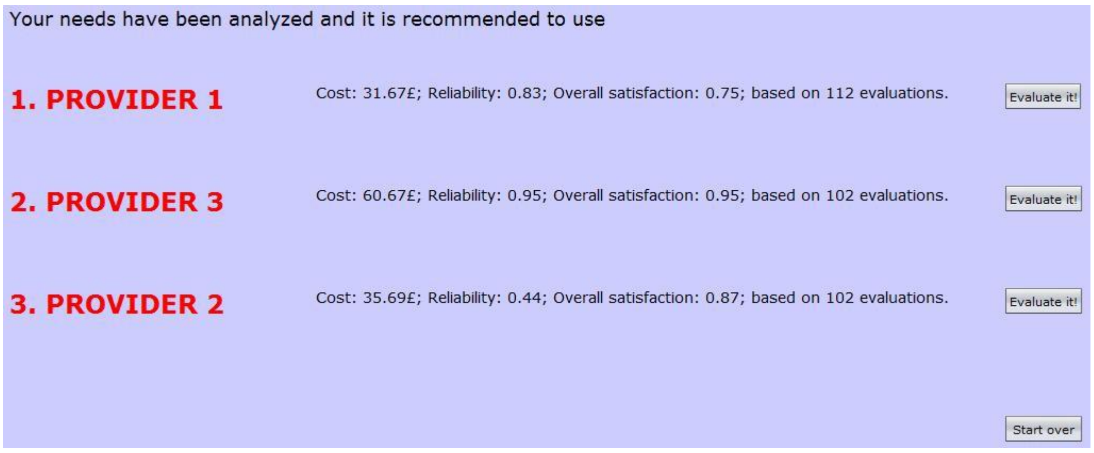

Your needs have been analyzed and it is recommended to use

1. PROVIDER 1 Cost: 7.27£; Reliability: 0.83; Overall satisfaction: 0.75; based on 112 evaluations.

2. PROVIDER 2 Cost: 8.09£; Reliability: 0.44; Overall satisfaction: 0.87; based on 102 evaluations.

3. PROVIDER 3 Cost: 12.8£; Reliability: 0.95; Overall satisfaction: 0.95; based on 102 evaluations.

Figure 3.4: Results for the first case of scenario (screen shot)

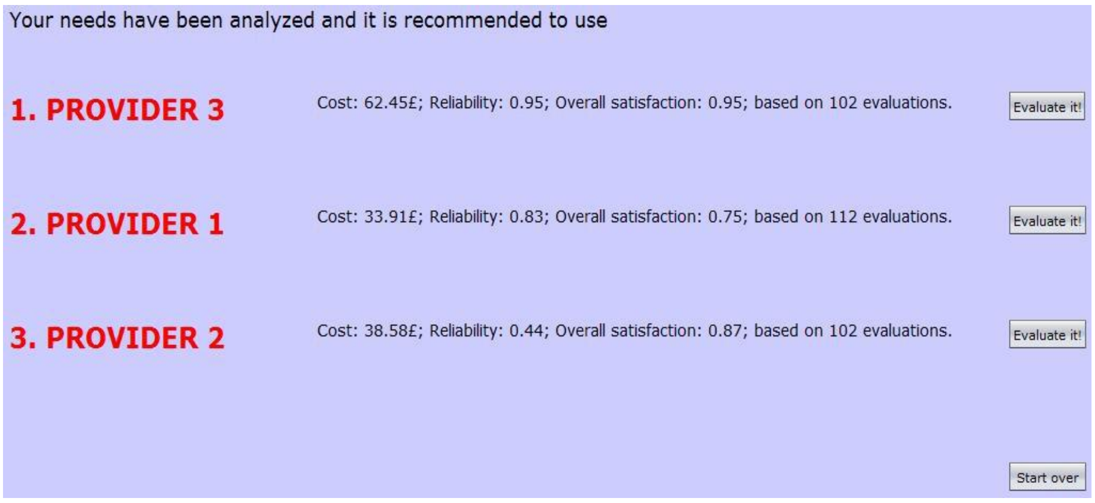

Your needs have been analyzed and it is recommended to use

1. PROVIDER 1 Cost: 31.67£; Reliability: 0.83; Overall satisfaction: 0.75; based on 112 evaluations.

2. PROVIDER 3 Cost: 60.67£; Reliability: 0.95; Overall satisfaction: 0.95; based on 102 evaluations.

3. PROVIDER 2 Cost: 35.69£; Reliability: 0.44; Overall satisfaction: 0.87; based on 102 evaluations.

Figure 3.5: Results for the second case of scenario (screen shot)


Your needs have been analyzed and it is recommended to use

1. PROVIDER 3 Cost: 62.45£; Reliability: 0.95; Overall satisfaction: 0.95; based on 102 evaluations.

2. PROVIDER 1 Cost: 33.91£; Reliability: 0.83; Overall satisfaction: 0.75; based on 112 evaluations.

3. PROVIDER 2 Cost: 38.58£; Reliability: 0.44; Overall satisfaction: 0.87; based on 102 evaluations.

Figure 3.6: Results for the third case of scenario (screen shot)

The results for the three cases presented in the scenario are shown in Figures 3.4, 3.5 and 3.6 respectively. These results are for the cost and QoS inputs shown in Table 3.2.

3.4.4.1. Evaluation of the Information Broker and their Implications

The evaluation of the broker revealed that it was capable of selecting the cheapest provider fulfilling the user's requirements. The Information broker experiment was successful as the broker was not only capable of effectively managing provider's information it was also capable of processing user's queries for Cloud resources. Therefore the information storage and selection processes in the envisioned system would be based on those used by the Information broker.

The key limitation of the Information broker was in regards to its functionality; which was limited to sharing of the information. The envisioned system should not only provide the information regarding Cloud resources but also be able to acquire and setup those resources. Other limitations include a lack of QoS information sources as there were only two QoS parameters. The only source of QoS information for these parameters was user's reviews. This limitation could be addressed by introducing other sources of information such as usage logs. The Information broker was implemented as a desktop based software application limited to Windows and Linux platforms therefore access was limited. The proposed solution should be platform independent and globally accessible.

3.5. Envisioned requirements

The requirements for the envisioned system were identified through literature, interviews with experts and the Information broker experiment. These are distributed into research and system requirements both of which should be fulfilled by the envisioned system.

3.5.1. Research Requirements

The research requirements are identified by the experts and are part of their vision of the envisioned service.

(1) Reduction in cost is the foremost requirement. This includes reduction in upfront and operational costs.

(2) Means to measure QoS associated with the Cloud providers. As Cloud providers offer best efforts; historic information regarding their performance would be crucial to any selection decisions.

(3) Instant access to any amount of resources at any time of the day. It was highlighted that in the current practice, invariable waiting times are associated with access to HPC resources.

(4) Access to different resource sizes such as small, medium, large and very large. As the user's requirement vary over the course of their research there should be adequate resource sizes to meet those requirements.

(5) Ability to increase or decrease the size of available resources. The instant increase in amount of available resources would help cope with surges in demand. The ability to decrease available resources would remove resource underutilization thus helping with the cost.

3.5.2. System requirements

The system requirements are associated with the development of the prototype for the envisioned solution. These were identified mainly through the interviews and in part by the Information broker experiment.

(1) The envisioned system should provide mechanisms for effectively managing the Cloud provider's information. This includes storing functional and QoS information and updating provider's QoS at regular intervals. The provider's QoS information should be updated automatically without any human interference.

(2) It should be able to accurately and efficiently process user's queries for Cloud provider's resources. The search interface should be simple and the search process should be fully automatic. The selection process should return the cheapest provider's VM that meats the user's requirements.

(3) The system should be capable of automatically setting up and running the required Cloud resources. The system should transfer the control of the selected resources to the user in a seamless fashion.

(4) The envisioned system should ensure a standard way of describing the Cloud providers. At the moment each provider has its own vocabulary which can be confusing to the user.

(5) The selection process should be able to handle large numbers of user requests simultaneously. This would require mechanisms for reducing the processing times without affecting the accuracy of the search process.

(6) Communication between the Information Management, Selection and Run should be seamless. This would require the system to ensure mechanisms for communication between semantic and non semantic services.

(7) The system should automatically track the availability of the different services on offer from the Cloud provider's. If a service is unavailable it should not be included in the search process.

3.6. Summary

This chapter has described the requirements for the envisioned system. These were identified by literature review of the current practice, interviews with the experts and an Information broker experiment. The literature review and the interviews investigated the current practice and identified the general requirements from the perspective of experts. The Information broker experiment was designed to highlight the system requirements which are concerned with the technical aspects of the envisioned system. For example the Information broker experiment identified that the envisioned system should not be a desktop based application as these are platform dependent and have limited accessibility.

The expert requirements were from the lab director and computational modeller's perspective. The lab director's requirements were focused on cost, security and reliability while the computational modeller requirements included instant access and variable sized machines. The system requirements identified that the envisioned service should accurately and efficiently select the best (cheapest) Cloud provider's VM; fulfilling the users functional and QoS requirements.

Chapter 4
Architecture of the QaComPS

The aim of this research as described in chapter 1 was to identify the best (cheapest) Cloud provider's Virtual Machine (VM) fulfilling the users functional and QoS requirements. This chapter describes the architecture of the Quality aware Computational Cloud Selection (QaComPS) service. The QaComPS is the proposed solution for addressing the user's queries for Cloud provider's VMs.

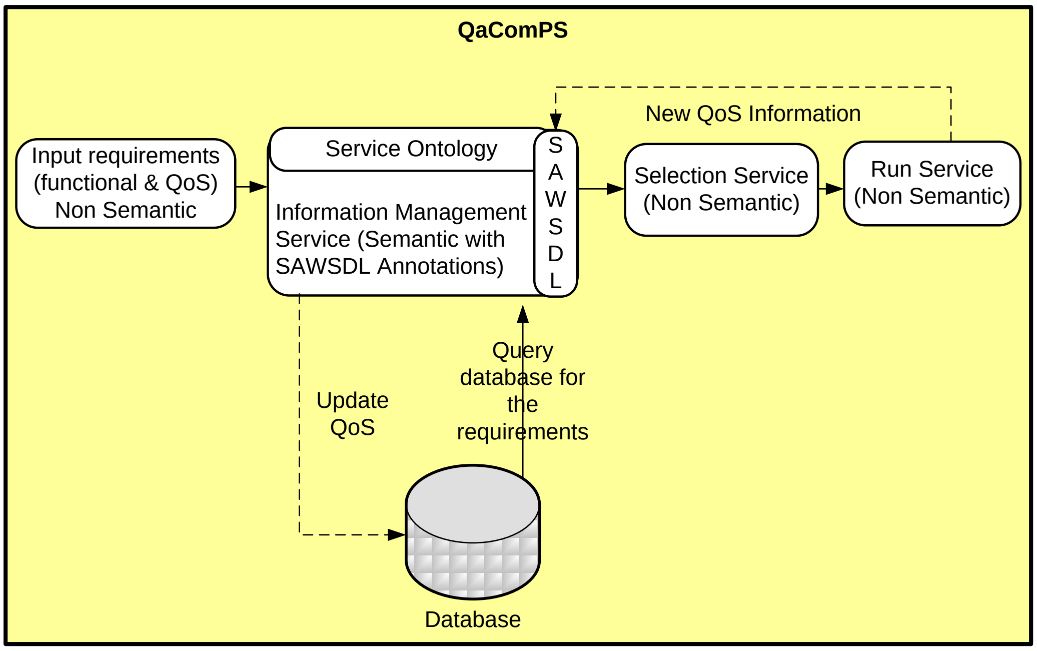
The chapter is divided into five sections where the first section presents an overview of the QaComPS. Section 4.2 describes the inputs requirements. This contains the description of the service interface which is used for inputting the user requirements. Section 4.3 describes the Information Management service. In the next section the selection service is described; this contains a step by step description of the selection process. This is followed in section 4.5, by the description of the Run service which is called upon by the selection service for setting up and running the selected Cloud provider's VM.

4.1. Overview of the QaComPS

An overview of the service architecture is shown in figure 4.1. The selection process takes place in five stages. These are input user requirements; query the Cloud provider's database; selecting the best VM; setting-up/running the selected VM; and updating the database.

- *Input User Requirements:* The user's functional and QoS requirements are inputted through the service interface. The functional requirements describe the resources being offered while the QoS describe quality requirements.

- *Information Management Service*: This is an internal, semantic service that adds; updates; and deletes Cloud provider's information. This information is required for processing user's queries. The service dynamically queries the database for every user request and passes the results onto the selection service. This service is managed by the service manager who is responsible for adding, updating and deleting provider's information.


QaComPS
New QoS Information
Input requirements (functional & QoS) Non Semantic
Service Ontology
Information Management Service (Semantic with SAWSDL Annotations)
SAWSDL
Selection Service (Non Semantic)
Run Service (Non Semantic)
Update QoS
Query database for the requirements
Database

- The solid lines represent the flow of information
- The dotted lines represent the steps that occur at the end of the transaction

Figure 4.1: Overview of the QaComPS Architecture

- *Selection Service*: This is an internal service; responsible for processing the user's queries for Cloud provider's VMs. The service has access to the Information Management service.

- *Run Service:* This is an internal service which can only be invoked by the selection service. It sets up and starts the selected Cloud provider's VM.

- Information Database: The services are underpinned by Information database. The database is accessible through the Information service. The database is updated at regular intervals as new QoS information is available after each transaction.

4.2. Input User Requirements

The user inputs to the QaComPS are distributed into functional and Quality of Service (QoS) inputs. The functional information is a description of the physical resources on offer from a Cloud provider. The QoS information is a description of the provider’s performance.

4.2.1. Functional Inputs

The functional inputs are amalgamated as a single input called the VM size. The VM size varies with respect to the amount of hardware and software resources associated with the VM. Typically four VM size are available through a Cloud provider with some such as Amazon EC2 offering as many as 17 different VM sizes (Amazon 2012).

Table 4.1 contains the typical functional values and the expected cost. These are based on four actual Cloud providers namely Amazon, FlexiScale (FlexiScale 2012), GoGrid (GoGrid 2012) and Rackspace (Rackspace 2012). The functional values for four providers were mapped onto the proposed model using flexible ranges. For example a small VM for AmazonEC2 has 1.2 GHz of processing while the same for FlexiScale has 1 GHz of processing. The small VM for the proposed model has processing between 1 and 1.2 GHz. This flexible range means that the mapping is not fixed to the specification of a single Cloud provider but represents all of the providers.

	Monthly cost ($)	**Memory (GB)**	**Processing (cores)**	**Storage (GB)**
Small	Up to 100	2.00	1	160
Medium	Up to 350	4.00	4	500
Large	Up to 500	8.00	8	800
Very Large	Up to 1000	16.00	16	1700

Table 4.1: QAComPS Cost Model

The processing unit used by the QaComPS is "cores" with each core having between 1.0 and 1.2 GHz of processing.

There are two forms of storage on the Cloud; persistent and non-persistent both of which are measured in GB. The persistent storage is permanent and independent of the VM while the non persistent storage is lost once the VM stops functioning. This research is concerned with the non-persistent storage; typical values for which are shown in table 4.1.

The VM cost does not cover bandwidth which represents the data transfer (upload/download) between the VM and the user's machine. This is measured in GBs and charged exclusively per use. The bandwidth is not related to the VM size therefore it is not included in table 4.1. The typical cost of bandwidth is 10-15 cents/GB.

The software resources associated with the VM are Operating System (OS), firewall and the hypervisor. Options for OS include Windows, Linux, and Solaris out of which only Windows is charged while the others are free. The firewall is by defaulted included with the VM in most cases however some providers such as

FlexiScale do charge it explicitly (FlexiScale 2012). The hypervisor is the technology used for creating VMs by virtualization. Two of the most popular hypervisor software are Xen and VMware (Rimal, Choi et al. 2009).

4.2.2. QoS Inputs

The Information broker described in section 3.3 of the previous chapter had two QoS parameters namely reliability and reputation. The values for these parameters were based entirely on the user's evaluations of the provider. During the user interviews it was identified that the lab in-charge was particularly interested in cost and security. Therefore QoS parameters managed by the QaComPS also include cost, security along with reputation and reliability.

In practice a larger set of QoS would be required but this is outside the scope of this thesis. Availability and performance are used extensively for Grid based solutions (Buyya, Abramson et al. 2000; Foster, Freeman et al. 2006; Montella 2007). In the context of this research performance is considered as a part of the reliability as there is a maximum limit of time associated with running the job. If the job is not completed within that time the reliability is lowered. The reputation ratings given by the user are also indicative of the Cloud provider's performance. During the user's rating process a question is asked regarding the performance of the provider. Availability is not explicitly monitored by the proposed solution. The exclusion is due to the definition of Cloud computing which states that access to resources should be on-demand (Peter Mell 2011). Availability is however implicitly included as the service ontology states that each provider should have at-least one VM available to be considered an active provider.

The QoS inputs consist of a quality input which describes the required quality on a scale of low, medium and high. The other QoS inputs are the QoS weights for the four parameters. These determine the relative importance of each parameter. The weights are inputted on a scale of one to nine with one being the lowest.

4.2.3. Service Interface for Inputting Information

The QaComPS service interface enables communication between the user and the QaComPS service. The design for the QaComPS service interface is shown in figure 4.2. The interface enables the users to inputs resource requirements while it enables the lab manager to manage the service.

The user database contains user's credentials which are required for managing access and tenancies (required for billing). Managing access and tenancies are not under the scope of this research therefore are not pursued further.

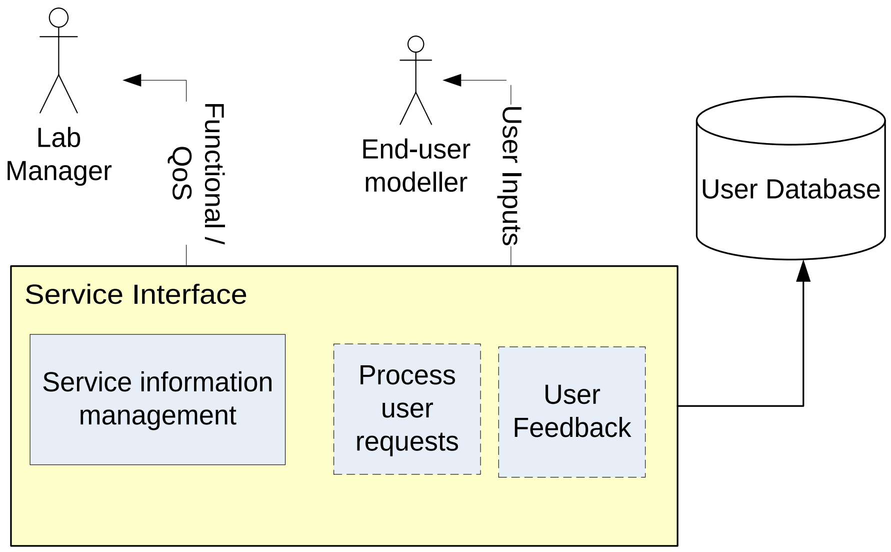


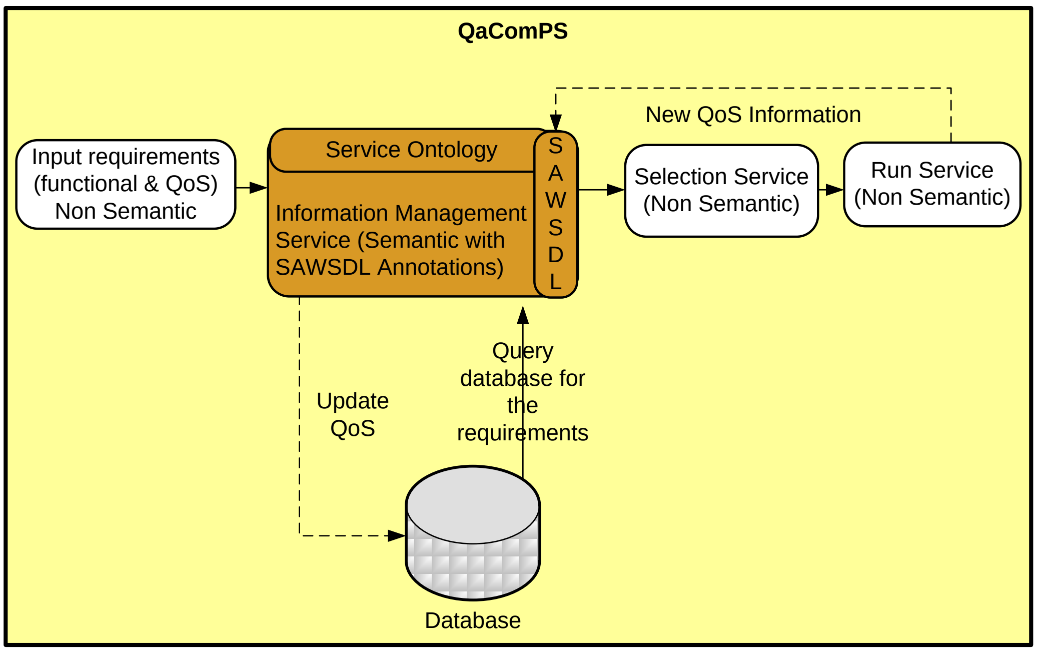
Figure 4.2: QaComPS service interface

The activities associated with the user are shown with a dotted line; these are processing user request and feedback. The interface processes user's requests by converting the inputs into process-able information. This information would be passed onto the Information Management service as it would be used for querying the database. Like the query information the feedback information is also passed onto the Information Management service.

The interface also authorizes access to the lab manager. The lab manager requires access to the Information Management service for adding, updating or deleting Cloud provider's information.

4.3. QAComPS Information Management Service

The effective management of Cloud provider's information is of prime importance to this research. As each step of the QaComPS selection process is either directly or indirectly dependent on the Cloud provider's information. The Information broker described in (see chapter 3.3) forms the basis for the QaComPS Information Management service.


QaComPS
New QoS Information
Input requirements (functional & QoS) Non Semantic
Service Ontology
Information Management Service (Semantic with SAWSDL Annotations)
S A W S D L
Selection Service (Non Semantic)
Run Service (Non Semantic)
Update QoS
Query database for the requirements
Database

- The solid lines represent the flow of information
- The dotted lines represent the steps that occur at the end of the transaction

Figure 4.3: QaComPS Information Management Service

This section is sub divided into six sub sections where the first section describes the QoS metrics (model for processing QoS information). The next section describes the service ontology which includes ontology descriptions for the Cloud provider, QoS and filter. The third sub section describes the storage of information as RDF. Section four describes the filter which queries the Cloud provider's information. The fifth section describes the semantic aspects of the service and annotations used for communication with other non-semantic services. The final sub section describes the information update process.

4.3.1. QoS Metrics

The four QoS parameters are measured using four QoS metrics namely: Reliability R_l (p), Reputation R_p (p), Security S (p), and Cost C (p) (where p stands for provider). The QoS information is measured on a scale of 1-9 where one represents the lowest level of QoS while nine represents the highest level of QoS.

Reliability: R_l is the probability that a provider's service would succeed in completing the execution of a job. Success rate is computed using data from the past invocations of the service. Following expression is used for computing reliability:

$$R_l = N(p)/K$$

N is the number of times a provider's VM has successfully completed a job within the maximum expected time frame. *K* is the total number of invocations. Closely related to reliability is the availability however this does not apply to the Cloud providers as these offer on-demand access. Therefore availability is not required for this research.

Reputation: R_p is the reputation of the Cloud provider. This is a measure of the trustworthiness of the provider. It's a posterior estimation based on historic end-user experiences. The reputation, is measured with the user feedback; collected from the user's experience with the service. The user rates the service for performance, customer support, value for money and the overall experience. The reputation ratings are updated using:

$$NewRp = OldRp + \left(\frac{(Pr+CSr+VMr+Oer)}{4} - 5\right)$$

Pr: Performance rating, CSr: customer support rating, VMr: value for money rating, Oer: Overall experience rating

The mean for the four rating parameters is subtracted by five as five indicates the middle value. A value above five is a positive rating while a value below it is a negative rating.

Security: The QoS security is a rating for the security offered by a Cloud provider. A number of parameters affect the security of a Cloud system. In this work, for illustrated purposes, two key parameters of security are described namely access security and the data security (Mather, Kumaraswamy et al. 2009).

The access security monitors the level of security at the access level. Each provider has a different approach to authentication some use only passwords while others use security certificates and some use Role Based Access Control (RBAC) (Ferraiolo, Kuhn et al. 2007). For example Amazon offers each VM a separate security key which means in cases of access security lapses only a single machine is compromised.

The security rating is monitored by analyzing the authentication method in use by the provider and tracking for security breaches at this level. The other factor for the security rating is the data security which concerns data being compromised. It is very difficult to identify this sort of security failure and the only source in this case is to use the literature. Security rating is calculated using:

$$Sec = \frac{(60as+40ds)}{100}$$

“as” is the access security while “ds” is the data security; sixty and forty are the respective percentages they make-up out of the total security. The access security is given a higher rating as it is a cause of data security lapses.

Cost: The physical cost (what a user pays) of a VM is translated into the cost QoS using:

$$Cost = (\alpha \ x \ C_l \ + \beta \ x \ Dt) / \ (\alpha + \beta)$$

where C_l represents VMs memory (GB), processor (virtual cores), and non-persistent storage. D_t is the data transfer rate and α, β are constants which represent the ratio of resource and data transfer to the overall cost.

$$MeanQoS = (2Qc + (QR+QP+Qs))/4$$

QC=Cost QoS, QR=Reliability, QoS, QP=Reputation QoS and Qs=Security QoS. MeanQoS rating describes the average QoS on a scale of one.

The MeanQoS which is defined above; it is used during selection when two providers have the same values, for the required QoS.

4.3.2. Service Ontology

The service ontology is a formal description of the Cloud providers; it includes the functional and QoS information. Each concept in the ontology is described as a Class. The QaComPS service ontology has three sub-ontologies namely Provider, QoS&Rank and Filter; these are shown in figure 4.4.

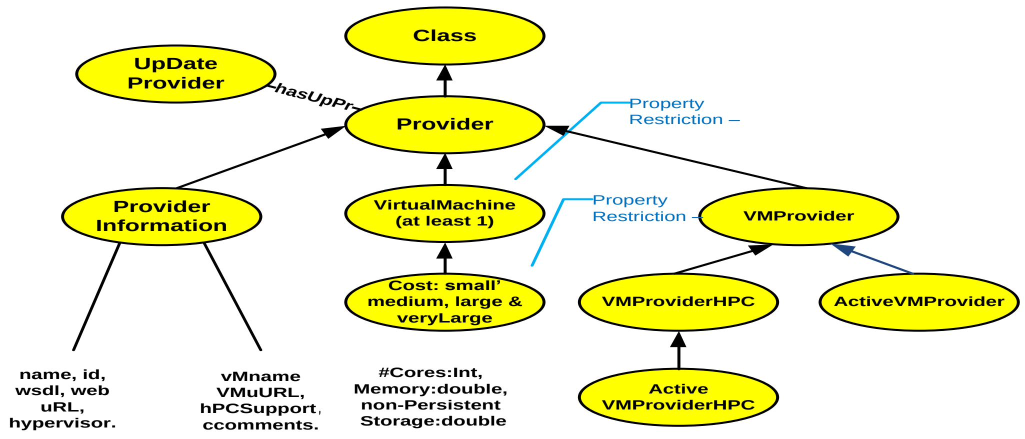

Class
UpDate Provider
hasUpPr
Provider
Property Restriction
Provider Information
VirtualMachine (at least 1)
Property Restriction
VMProvider
Cost: small' medium, large & veryLarge
VMProviderHPC
ActiveVMProvider
Active VMProviderHPC
name, id, wsdl, web uRL, hypervisor.
vMname VMuURL, hPCSupport, ccomments.
#Cores:Int, Memory:double, non-Persistent Storage:double

Figure 4.4a: Provider's ontology

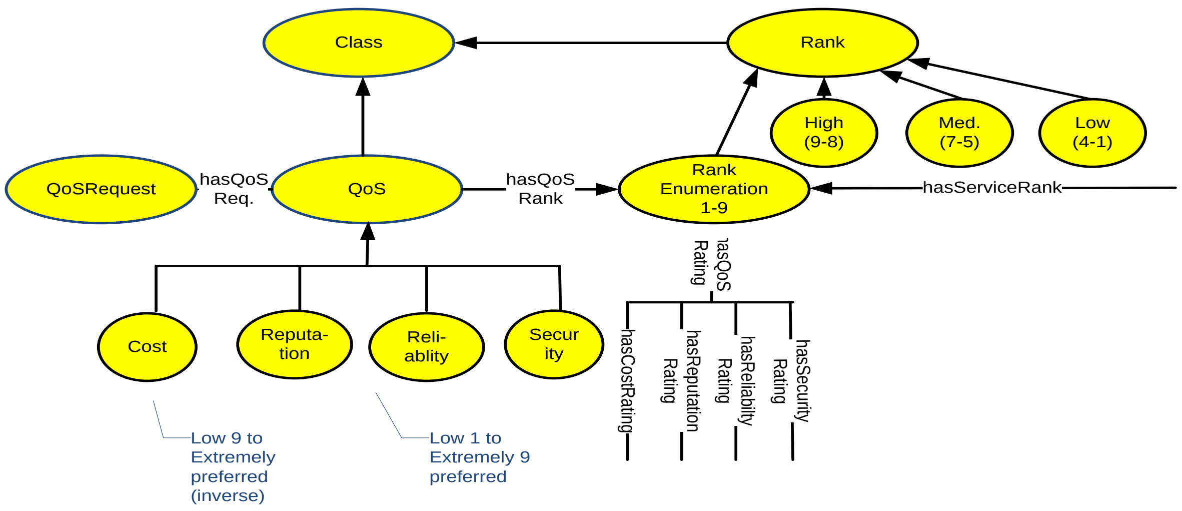


Figure 4.4b: QoS Ontology

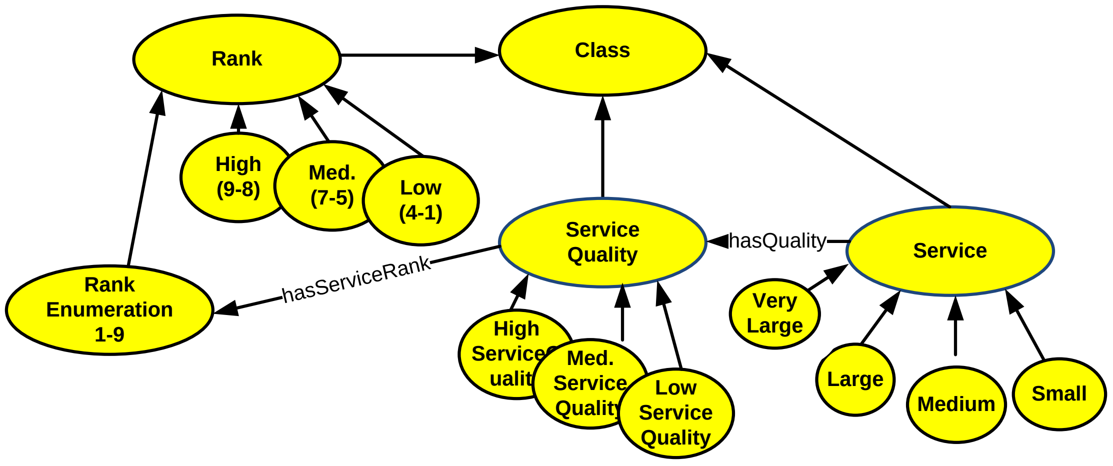


Figure 4.4c: Filter Ontology (fragment of Rank class is repeated)

The Provider's ontology describes the Cloud provider and the associated information as shown in Figure 4.4a. The top class is the Provider and a sub-class models the number of VMs offered by the provider. The cost class captures the four VM sizes as each has a varying cost. An OWL2 property restriction is used to ensure that there is at least one VM instance for each provider. Instances are members of the class, for example AmazonEC2_Small is a VM on offer from Amazon. In simpler terms a VM is a concept while the instance of a VM is a physical entity which can be allocated to a user. The VMProvider class models the type of VM and whether it's active or inactive. An UpDateProvider class describes the update process.

The QoS and Rank ontologies are based on the QoS model which consists of Cost, Reputation, Reliably and Security classes. It uses an OWL2 rank enumeration giving the meaning to the ranking. The data properties are used for storing associated provider data. The second part of the Rank class is shown in figure 4.4c.

The last part of the ontology is described in classes associated with filter these are Rank, ServiceQuality and Service. Part of the Rank class is to devise the QoS ratings into three classes namely High (8-9), Medium (5-7) and Low (1-4).

The interesting part of the ontology is the use of two classes ServiceQuality and Service. These classes are linked by the hasQuality object property. The purpose of this is to match the service quality (divided into high, medium and low) with four service levels (small, medium, large and very large).

The ServiceQuality is constructed using OWL2 union property restriction to describe:

$$UQoS = C \,\&\, R_p \,\&\, R_l \,\&\, S$$

where C, R_p, R_l & S are described 4.2.1.2.

The meaning assigned to each service quality class is as follows: HighServiceQuality is UQoS ≥ 8; MediumServiceQuality is UQoS ≥ 3 and < 5; and LowServiceQuality = UQoS < 5. The semantic Web statements are shown in figure 4.4c.

4.3.3. Storing Semantic Information

The information is stored in the database using a RDF format. Provider entries made by the RDF component are based on the mapping between the provider and the provider ontology. This mapping ensures that all the providers conform to a single vocabulary which simplifies the provider's descriptions as currently each Cloud provider has its own vocabulary.

The RDF information would use meaningful naming conventions for improving the querying process. The prefix in the name of an RDF file would be the name of the provider, while the suffix would be the VM size. For example AmazonEC2_Small where AmazonEC2 is the provider while small is the VM size.

4.3.4. Filtering Providers

The users functional and QoS inputs are passed onto the filter. The filter improves the efficiency of the selection process by reducing the number of potential providers. The filter ontology ensures that the provider's QoS information is stored in a specific way to assist the filtering process.

It is concerned with two aspects of the selection process. Firstly it is concerned with matching the user's functional requirements to the functional properties of the Cloud service providers. Secondly matching the user's QoS requirements with the provider's QoS. The matches should be exact for the first case; for example if a user requires small VM only small VMs should be part of the selection with the other VM sizes being excluded. In case of QoS matches; the user's requirements should be treated as minimum requirement. For instance requests for low QoS can return results for both medium and high QoS; while those for high QoS should return results for only high QoS.

The filter queries the functional information with the help of the naming convention. Example for such a query is given in figure 4.5.

```
SELECT * FROM providers WHERE provider_name like '%Small';
```

Figure 4.5: Query for extracting small sized VMs

The query for extracting all the small VMs is shown in figure 4.5. The % sign means any number of characters so the query would extract all the information regarding providers who has the word "Small" at the end of their names.

```
PREFIX QoS=http://localhost:8080/Ontology/QoS#

PREFIX Filter=http://localhost:8080/Ontology/Filter#

PREFIX Provider=http://localhost:8080/Ontology/ Provider #

SELECT ?ID ?Name

WHERE{ ?p Provider: hasID ?ID .

?p Provider: hasName ?name.

?q QoS: hasReliability ?Rel. FILTER(?Reliability >=  queryParam[1]).

?q QoS: hasReputation ?Rep. FILTER(?Reputation >=  queryParam[2]).

?q QoS: hasSecurity ?Sec. FILTER(?Security >=  queryParam[3]).

?q QoS: hasCost ?Cost. FILTER(?Cost >=  queryParam[4]).}
```

Figure 4.6: SPARQL query for extracting providers with user's required QoS

The QoS information is queried by SPARQL query. It identifies the list of providers that match the user's QoS requirements. The SPARQL query shown in figure 4.6 uses three ontologies. The query filters providers that have four QoS values; greater than or equal to that of the users requirements. The provider id and name are used for identifying the provider. The field queryParam contains user's QoS requirements which are matched against the provider's QoS.

4.3.5. SAWSDL annotations for Communication between Services

The communication channel between the semantic Information Management service and other non-semantic service is based on SAWSDL semantic annotations. These enable the sharing of descriptions and transfer of information between semantic and non-semantic sources.

4.3.5.1. Model References for Sharing Descriptions

SAWSDL “Model References” are used for sharing the entity level descriptions of information between services. Example of a model reference is given below:

```
<xs:element name=”Reliability”

sawsdl:modelReference="http://localhost:8080/Ontology/QoS#Reliability">
```

The above example contains the SAWSDL annotation for the description of the Reliability class. The Reliability class is part of the QoS ontology and is available and the given URI. The underlying technology for writing SAWSDL annotations is XML. This information can be parsed by any (semantic/non semantic) web services as these are written in WSDL; which is an XML based description of the service.

4.3.5.2. Lifting/Lowering Schema Mapping for Transfer of Information

SAWSDL annotations use lifting and lowering schema mapping for transferring information between semantic and non-semantic services. Both use querying languages for extracting information. The lifting schema mapping uses XQuery (Boag, Chamberlin et al. 2003) which is a query language for querying XML sources. The lowering schema mapping uses SPARQL which is used for querying semantic sources.

Lifting schema mapping enables the Information Management service to receive the QoS update information from non-semantic sources. Lifting schema mapping was achieved by creating XQuery which is an xml based scripts. An example for the lifting of non semantic information is given in figure 4.7. The example queries the non-semantic information source for the reliability rating. The type for the information would be double; number with decimals.

```
<po:QoS>
  <po:Reliability>
  <po:hasRelRating
rdf:datatype="http://www.w3.org/2001/XMLSchema#double">{      fn:integer(/)
}</po:hasRelRating>
…
</po:QoS>
```

Figure 4.7: Lifting Schema Mappings

The lowering schema mappings were used for transferring semantic information from the Information Management service to the selection service. The transferred information is queried from the database and is required for the selection process. Lowering schema mapping uses a SPARQL query to extract information from the database and passes it onto the non-semantic selection service. This mapping uses SPARQL query to map the semantic information onto a non-semantic web service.

4.3.6. Information Update

The Information Management service has provisions for both manual and automatic updating of the information; depending on the available sources of information. Manual updates would be performed by the lab managers and is used only when there are no means of updating the information automatically.

The functional information is available through the Cloud providers and will be updated only when the providers make any changes. Automation of functional requirements is possible in one of the two ways either through the use of crawlers or through the use of intelligent agent to update the information. Both of these methods are out of the scope of this thesis. (Nelson, Smith et al. 2006).

The QoS information for reputation would be received through user's reviews of the service. The reliability and security information would be updated through the reviews and usage logs which indicate events such as start, stop and any failures. The cost QoS is difficult to manage automatically as it is dependent on physical cost (money paid for the VM) which is managed by the Cloud provider.

The first step for updating information is to identify whether the given information conforms to the ontology descriptions for Cloud provider and QoS. After that the specified provider's RDF information is updated. The service maintains a separate RDF for each of the provider's VMs on offer.

4.4. QaComPS Selection Service

This research proposes a novel QaComPS selection service. This service receives Cloud provider's information from the filter (see 4.3.4). The selection process involves ranking the list of available providers and selecting the best provider.

4.4.1. Architecture of the QaComPS Selection Service

The service has a user interface for communicating with the users and a service interface for communicating with the Cloud providers (see section 4.1.3). The service interface uses WSDL (Moreau, Chinnici et al. 2006) or Application Programming Interface (API) depending on the access mechanisms supported by the Cloud provider.

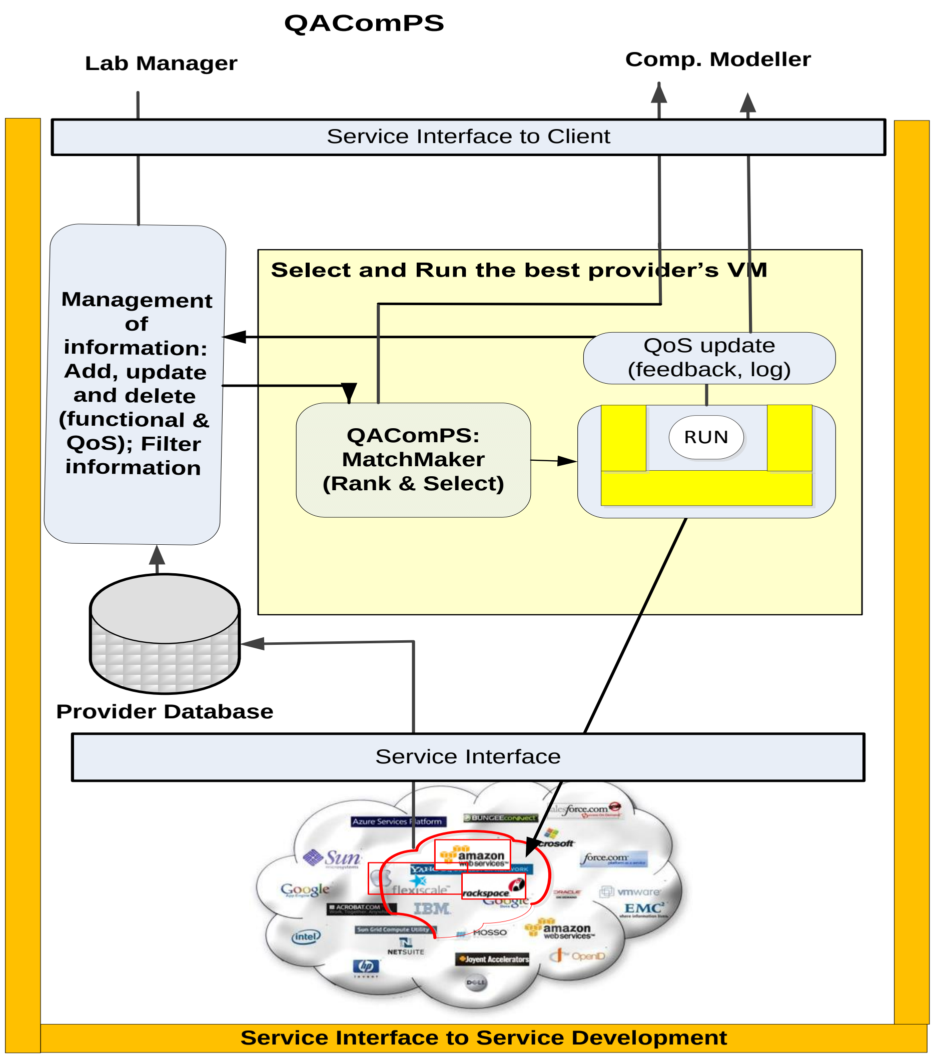

QAComPS
Lab Manager
Comp. Modeller
Service Interface to Client
Management of information: Add, update and delete (functional & QoS); Filter information
Select and Run the best provider's VM
QoS update (feedback, log)
QAComPS: MatchMaker (Rank & Select)
RUN
Provider Database
Service Interface
Service Interface to Service Development

Figure 4.8: QaComPS Architecture

The QaComPS selection service is shown in figure 4.8. The selection process is initiated by receiving user inputs through the input interface. The user inputs were described in section 4.1. The QoS inputs are measured on a scale of high, medium, or low. The QoS weights are also inputted which identify the relative importance of each QoS parameter through the user interface.

The novel feature of the proposed selection model is its treatment of the cost QoS metric. This is derived from the cost model which translates physical costs (what a user pays) into cost QoS (value for money). As the goal is to select the cheapest provider's VM the requirement for cost QoS is by default set to the highest level.

The architecture shows that the Information Management service can automatically acquire Cloud provider's information through the interface. However at the moment the availability of information is limited as the providers offer functional information through their web sites which require manual processing by the lab manager. Cloud providers whose entries are made by the Information Management service are shown by the red circle among the range of all the providers. The Run service has access to only those providers that are registered with the Information Management service.

4.4.2. MatchMaker

The matchmaker matches user's requests to the provider's resources in two stages. Inputs to the MatchMaker include the list of filtered providers. This information is the output of the filter component which uses the SPARQL query to lower the semantic mappings. The QoS requirements and weights are also required which are inputted by the user. The top three ranked results are passed onto the selection component for the final selection.

The first step for the matchmaking process is to rank the Cloud providers. This involves ranking the list of available providers with respect to their nearness to the user's requirements. The ranking is performed by using the Euclidean distance algorithm. The ranking process is based on QoS ratings of the provider's and the QoS requirements of the users. The final stage is the selection of the best (cheapest) provider's VM. Inputs to the AHP are the top three ranked providers. The selection process is based on the *Analytic Hierarchy Process (AHP*) (Saaty 2005).

The selection process matches the VMs against the user's requirements using the QoS levels and QoS weights (see section 4.2.2). The selection process returns the best (cheapest) provider meeting user's functional and QoS requirements.

4.4.2.1. Ranking Providers

The ranking step uses Euclidean distance algorithm to rank the list of available providers. Euclidean distance algorithm is used for calculating the distance between the user’s ideal provider and the available providers. The smallest distance means the best match and vice versa. The target being to find the provider’s VM with the minimum Euclidean distance.

The proposed method is based on the assumption that resources can be evaluated on the basis of their closeness to the user's requirements, taking into consideration the relative weight of each requirement. In mathematical terms, the closeness between two objects can be expressed by their Euclidean distance (ELEYAN, Amna et al. 2004). Geometrically this is a straight-line distance between two points, representing objects in m-dimensional space. Therefore, the best provider is the one that has the shortest distance from the given user's requirement, while the one with the farthest distance is the worst. All other providers can be ranked between these two extremes, with regards to the value of their Euclidean distance (Tomaž Klančnik 2009). The step by step process for the ranking of Cloud providers is given below.

Step 1: Convert QoS weight inputs to a pair-wise comparison matrix

A pair-wise QoS matrix for the QoS weights is created by reciprocating the inputted values. The pair-wise comparisons shown in matrix B are the reciprocal of matrix A (see following example). The diagonal of the matrix is always one.

The proposed selection service can accommodate the processing of any number of providers. However in an effort to keeping the given example simple the Cloud providers sample size is set to four.

Example:

User inputs for the four QoS parameters are as follows:

- Reliability is twice as important as reputation.
- Reputation is three times as important as security.

- Reliability is four times as important as security.
- Cost has the highest priority.

$$\text{Matrix A} = \begin{array}{c|c} \begin{array}{ccc} Rp & Rl & Sec \end{array} & \\ \begin{bmatrix} 1 & 0.5 & 3 \\ & 1 & 4 \\ & & 1 \end{bmatrix} & \begin{array}{l} Rp \\ Rl \\ Sec \end{array} \end{array}$$

$$\text{Matrix B} = \begin{bmatrix} 1 & 0.5 & 3 \\ 2 & 1 & 4 \\ 0.3333 & 0.25 & 1 \end{bmatrix}$$

Rp represents the reputation QoS, Rl is for reliability while Sec stands for security QoS.

Step 2: Calculate relative criteria weights

This requires adding all the values in a column and dividing each criterion by value sum. The criteria weights are calculated using the following equation.

$$\text{Wi}= \frac{1}{number\ of\ Crieteria} \left(\frac{QoS\ weight\ for\ crieterion\ i\ in\ colum\ 1}{sum\ of\ the\ entries\ in\ colum\ 1} + \frac{QoS\ weight\ for\ crieterion\ i\ in\ colum\ 2}{sum\ of\ the\ entries\ in\ colum\ 2} + \frac{QoS\ weight\ for\ crieterion\ i\ in\ colum\ 3}{sum\ of\ the\ entries\ in\ colum\ 3} + \ldots + \frac{QoS\ weight\ for\ crieterion\ i\ in\ colum\ n}{sum\ of\ the\ entries\ in\ colum\ n} \right)$$

The set of n relative weights is normalized to sum of one,

$\sum_{i=1}^{n} Wi = 1$, $W_i > 0$, i=1, 2, 3, … ., n.

Therefore the number of independent weights is (n-1)

Example:

$$\begin{array}{cc} \begin{bmatrix} 1 & 0.5 & 3 \\ 2 & 1 & 4 \\ 0.3333 & 0.25 & 1 \end{bmatrix} & \begin{array}{l} Rp \\ Rl \\ Sec \end{array} \end{array}$$

Sum of columns = [3.3333 1.75 8]

Reputation Weight = 1/3 X $(\frac{1}{3.3333} + \frac{0.5}{1.75} + \frac{3}{8})$ = 1/3 * 0.9607 = 0.3202

Reliability Weight = 1/3 X $(\frac{2}{3.3333} + \frac{1}{1.75} + \frac{4}{8})$ = 1/3 * 1.6714 = 0.5571

Security Weight = 1/3 X $(\frac{0.3333}{3.3333} + \frac{1.75}{1.75} + \frac{1}{8})$ = 1/3 * 0.3678 = 0.1226

$$Wi = \begin{bmatrix} 0.3202 \\ 0.5571 \\ 0.1226 \end{bmatrix}$$

Sum of QoS weights = 1

Step 3: Calculate denominator of normalizing equation

This step is concerned with the provider's QoS values. These values are normalized by using the following equation.

Denominator for normalizing equation Np = $\sqrt{\sum(Provider\ QoS\ Data)^2}$

Example:

$$\begin{array}{ccccl} \text{Prov1} & \text{Prov2} & \text{Prov3} & \text{Prov4} & \\ 9 & 5 & 2 & 3 & \text{Reliability} \\ 7 & 2 & 3 & 6 & \text{Reputation} \\ 8 & 4 & 1 & 5 & \text{Security} \end{array}$$

Provider's Reliability QoS: $\sqrt{81 + 25 + 4 + 9}$ = 10.9087

Provider's Reputation QoS: $\sqrt{49 + 4 + 9 + 36}$ = 9.8994

Provider's Security QoS: $\sqrt{64 + 16 + 1 + 25}$ = 10.2956

$$Np = \begin{bmatrix} 10.9087 \\ 9.8994 \\ 10.2956 \end{bmatrix}$$

Step 4: Normalize the performance matrix

This step involves normalizing the provider's QoS information. The normalized performance matrix (Pr) is calculated by dividing each value by the relative normalized QoS data; calculated in step 3.

$$\text{Pr}=\frac{\text{Provider 1 QoS for i}}{\text{Provider's NomalizedQoS for i}}\ \frac{\text{Provider 2 QoS for i}}{\text{Provider's NomalizedQoS for i}} \ldots \frac{\text{Provider n QoS for i}}{\text{Provider's NomalizedQoS for i}}$$

Example:

$$\text{Pr}=\begin{bmatrix} 9 & 5 & 2 & 3 \\ 7 & 2 & 3 & 6 \\ 8 & 4 & 1 & 5 \end{bmatrix} \ / \ \begin{matrix} 10.9087 \\ 9.8994 \\ 10.2956 \end{matrix} = \begin{bmatrix} 0.8250 & 0.4583 & 0.1833 & 0.2750 \\ 0.7071 & 0.2020 & 0.3030 & 0.6060 \\ 0.7770 & 0.3885 & 0.0971 & 0.4856 \end{bmatrix}$$

Step 5: Create weighted normalized performance matrix

This step involves multiplying user's normalized QoS requirements to the provider's normalized QoS. The weighted normalized performance matrix (Wpr) is calculated using the following equation.

Example:

Wpr = Wi(1)xPr(1) Wi(1)xPr(2) …… Wi(1)xPr(n)

Wi was calculated in step 2 while Pr was calculated in step 4.

$$\text{Wpr}=\begin{bmatrix} 0.3202 & 0.5571 & 0.1226 \end{bmatrix} \text{ X } \begin{bmatrix} 0.8250 & 0.4583 & 0.1833 & 0.2750 \\ 0.7071 & 0.2020 & 0.3030 & 0.6060 \\ 0.7770 & 0.3885 & 0.0971 & 0.4856 \end{bmatrix}$$

$$\text{Wpr} = \begin{bmatrix} 0.2642 & 0.1467 & 0.0587 & 0.0880 \\ 0.3939 & 0.1125 & 0.1688 & 0.3376 \\ 0.0952 & 0.0476 & 0.0119 & 0.0595 \end{bmatrix}$$

Step 6: Calculate relative Euclidean distance

The relative Euclidean distances are calculated using the following equation:

$$E_{\mathrm{j}} = \sqrt{\sum_{i=1}^{m}(\mathrm{Wpr}_{ij} - \mathrm{w}_i \mathrm{Qr}_i \,/ \sqrt{\sum_{i=1}^{n} \mathrm{Pr}_{ij}^{2}}\,)^{2}}$$

Where

j=1,2,....,n is the number of Cloud providers virtual machines;

i=1,2,…,m is the number of specification criteria (QoS values);

Qri represents an element of a vector r=(r1,r2,...rm) which represents the user's QoS requirements in terms of the i-th criterion.

Example

Provider, Rl, Rp, Sec, Euclidean Distance

[Prov4,   3.0, 6.0, 5.0, 0.2580]

[Prov2,   5.0, 2.0, 4.0, 0.2731]

[Prov1,   9.0, 7.0, 8.0, 0.2917]

[Prov3,   2.0, 3.0, 1.0, 0.6951]

The results show that Prov4 has the smallest distance to the user’s ideal provider while Prov3 has the largest distance.

4.3.2.2. Selection

The final selection is performed using the AHP algorithm (Saaty 2008). Inputs to the process are provider’s data and the user’s QoS requirements. The AHP hierarchy for selection of the best (cheapest) Cloud provider’s VM is shown in figure 4.9.

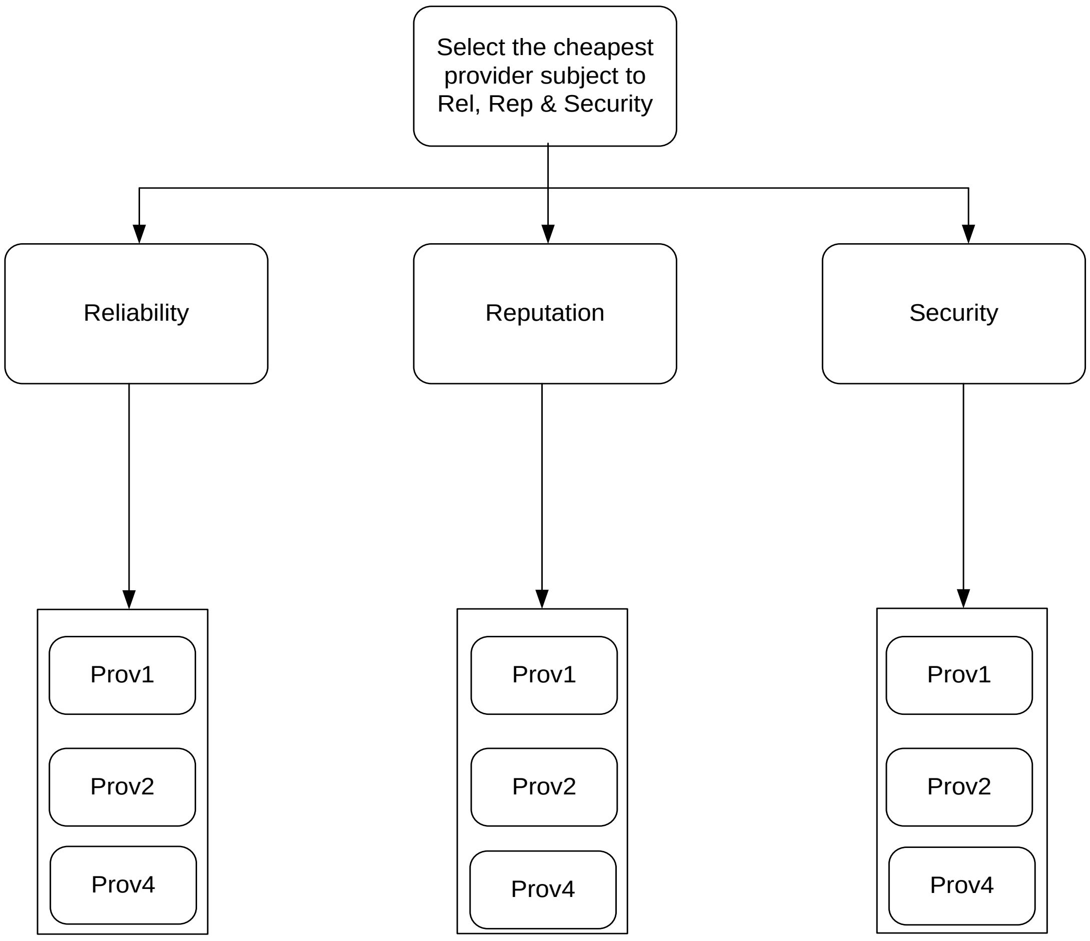


Figure 4.9: Hierarchy for selecting Cloud Provider VM

The hierarchy has three levels starting with the goal which is to select the cheapest provider fulfilling user's QoS requirements. This is followed by criterion for selection which is reliability, reputation and security while the cost is treated separately. The separate treatment of cost is due to its static QoS value set to the highest; as the target is to always identify the cheapest provider. The final level is for the alternatives which contain the list of potential Cloud providers.

Step 1: Convert user's inputs to pair-wise matrix

This was also the first step for the ranking process therefore is not repeated. The pair-wise comparison matrix for selection is given below:

$$\textit{QoS Weight Matrix} \begin{array}{c} \\ \\ \\ \\ \end{array}\begin{array}{ccc|c} \text{Rp} & \text{Rl} & \text{Sec} & \\ 1 & 0.5 & 3 & \text{Rp} \\ 2 & 1 & 4 & \text{Rl} \\ 0.3333 & 0.25 & 1 & \text{Sec} \end{array}$$

Step 2: Calculate the first Eigenvector

This consists of three sub steps; square the QoS weight matrix, calculate the sum total and normalize the matrix by dividing each row sum with the sum total.

Example

Square Weight Matrix Squared

$$\begin{bmatrix} 1 & 0.5 & 3 \\ 2 & 1 & 4 \\ 0.3333 & 0.25 & 1 \end{bmatrix} X \begin{bmatrix} 1 & 0.5 & 3 \\ 2 & 1 & 4 \\ 0.3333 & 0.25 & 1 \end{bmatrix} = \begin{bmatrix} 3 & 1.75 & 8 \\ 5.3332 & 3 & 14 \\ 1.1666 & 0.6667 & 3 \end{bmatrix}$$

Calculate the sum of the rows

$$\begin{bmatrix} 3 + & 1.75 + & 8 \\ 5.3332 + & 3 + & 14 \\ 1.1666 + & 0.6667 + & 3 \end{bmatrix} = \begin{array}{c} 12.75 \\ 22.3332 \\ \underline{4.8333} \end{array}$$

Sum the row totals: 39.9165

Normalize the row sum by the row totals to calculate the first Eigenvector

$$\begin{bmatrix} 12.75/39.9165 \\ 22.3332/39.9165 \\ 4.8333/39.9165 \end{bmatrix} = \begin{bmatrix} 0.3194 \\ 0.5595 \\ 0.1211 \end{bmatrix}$$

Step 3: Calculate the second Eigenvector

The second eigenvector is calculated in the same manner as the first eigenvector. This step involves squaring the updated (squared during the calculation of previous eigenvector) weight matrix. Calculating sum totals and normalizing.

Example

Square Weight Matrix Squared

$$\begin{bmatrix} 3 & 1.75 & 8 \\ 5.3332 & 3 & 14 \\ 1.1666 & 0.6667 & 3 \end{bmatrix} \text{X} \begin{bmatrix} 3 & 1.75 & 8 \\ 5.3332 & 3 & 14 \\ 1.1666 & 0.6667 & 3 \end{bmatrix} =$$

$$\begin{bmatrix} 27.6653 & 15.8330 & 72.4984 \\ 48.3311 & 27.6662 & 126.6642 \\ 10.5547 & 6.0414 & 27.6653 \end{bmatrix}$$

Calculate the sum of rows

$$\begin{bmatrix} 27.6653 & + & 15.8330 & + & 72.4984 \\ 48.3311 & + & 27.6662 & + & 126.6642 \\ 10.5547 & + & 6.0414 & + & 27.6653 \end{bmatrix} = \begin{matrix} 115.9967 \\ 202.6615 \\ \underline{44.2614} \end{matrix}$$

362.9196

Normalize the row sum by the row totals to calculate the second Eigenvector

$$\begin{bmatrix} 115.9967/\ 362.9196 \\ 202.6615/\ 362.9196 \\ 44.2614/\ 362.9196 \end{bmatrix} = \begin{bmatrix} 0.3196 \\ 0.5584 \\ 0.1220 \end{bmatrix}$$

Step 4: Compute the difference between the two Eigenvectors

Example

$$\begin{bmatrix} 0.3194 - 0.3196 \\ 0.5595 - 0.5584 \\ 0.1211 - 0.1220 \end{bmatrix} = \begin{bmatrix} -0.0002 \\ 0.0011 \\ -0.0009 \end{bmatrix}$$

The difference between the two eigenvectors is very minute (difference < = 0.001). If this was not the case the process would be repeated until the difference between the latest two eigenvectors is very minute.

Step 5: Relative importance of criteria

Reliability is the most important criteria while security is the least important as shown in figure 4.10.

Example

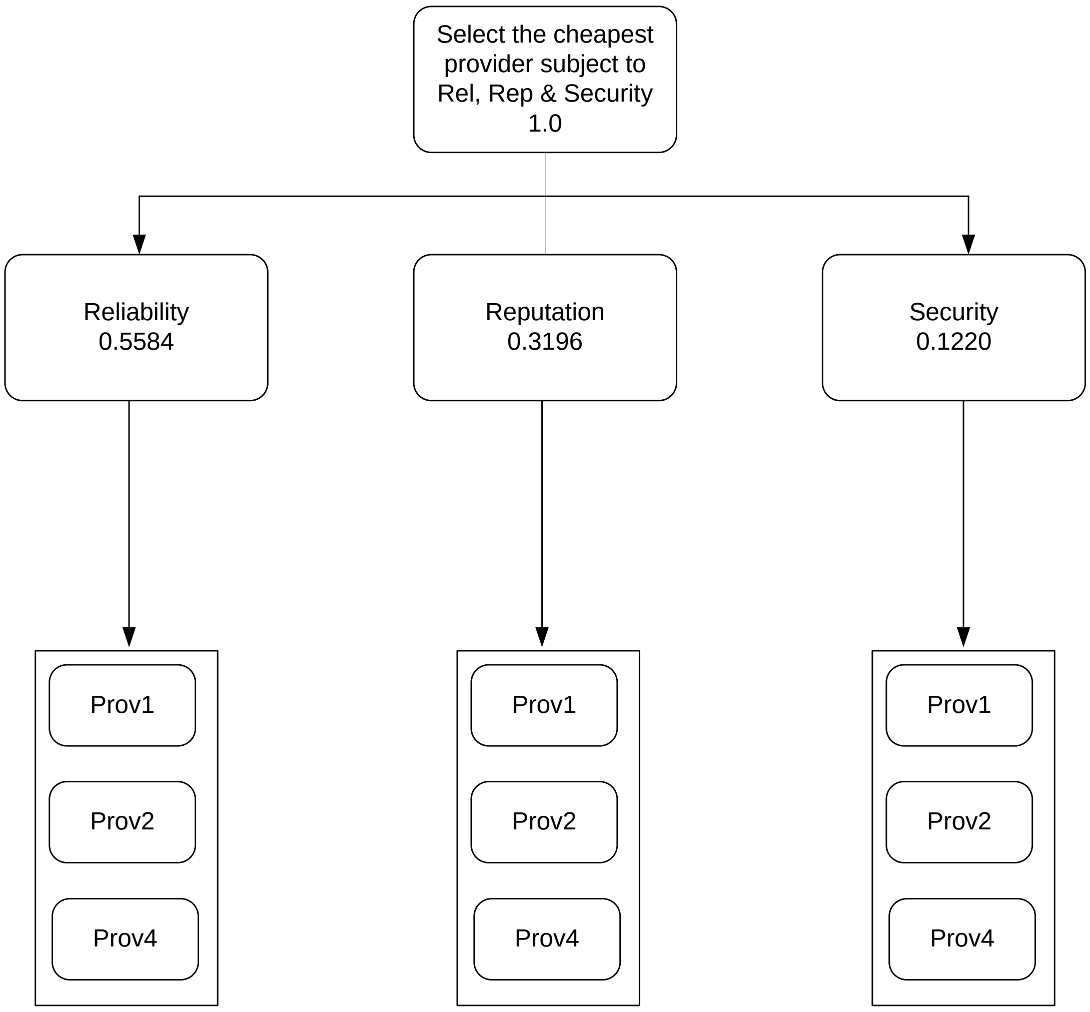


Figure 4.10: Relative importance of each criteria

Step 6: Calculate the normalized QoS for the three providers

This involves calculating the first eigenvector, second eigenvector and subtracting them. These steps would be repeated until the difference between the two is very small. Steps 2, 3, and 4 in this section describe this process.

Example:

Provider's QoS data

$$\text{Reliability:}\quad \begin{array}{c} \begin{array}{ccc} \text{Prov1} & \text{Prov2} & \text{Prov4} \end{array} \\ \begin{bmatrix} 1 & 1.8 & 3 \\ 0.5555 & 1 & 1.6667 \\ 0.3333 & 0.6 & 1 \end{bmatrix} \end{array}$$

$$\text{Reputation:}\quad \begin{array}{c} \begin{array}{ccc} \text{Prov1} & \text{Prov2} & \text{Prov4} \end{array} \\ \begin{bmatrix} 1 & 3.5 & 1.1667 \\ 0.2857 & 1 & 0.3333 \\ 0.8571 & 3 & 1 \end{bmatrix} \end{array}$$

$$\text{Security:}\quad \begin{array}{c} \begin{array}{ccc} \text{Prov1} & \text{Prov2} & \text{Prov4} \end{array} \\ \begin{bmatrix} 1 & 2 & 1.6 \\ 0.5 & 1 & 0.8 \\ 0.625 & 1.25 & 1 \end{bmatrix} \end{array}$$

Eigenvectors for the provider's QoS are given below:

$$\text{Reliability}\begin{bmatrix} 0.5294 \\ 0.2940 \\ 0.1764 \end{bmatrix} \quad \text{Reputation}\begin{bmatrix} 0.4986 \\ 0.1253 \\ 0.3760 \end{bmatrix} \quad \text{Security}\begin{bmatrix} 0.4705 \\ 0.2352 \\ 0.2941 \end{bmatrix}$$

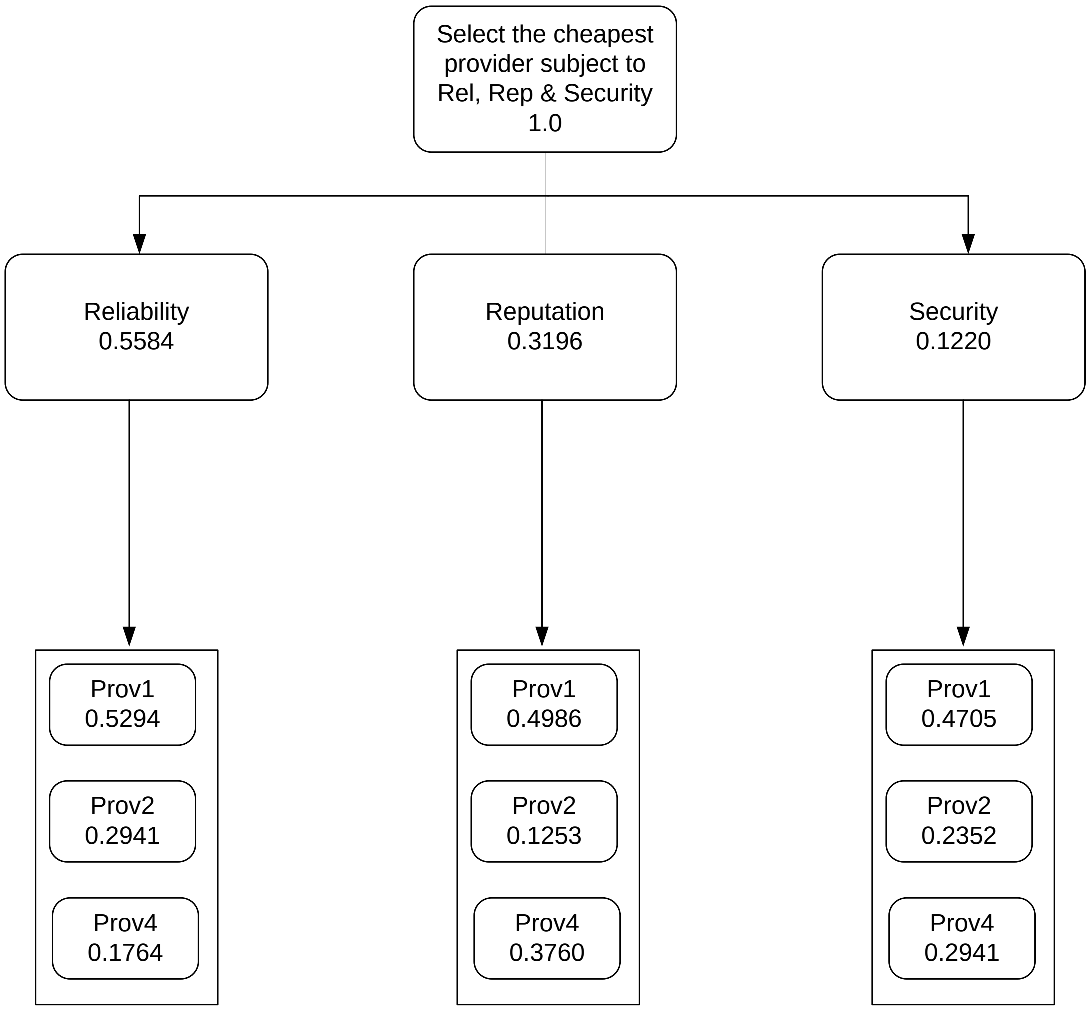


Figure 4.11: Relative importance of each criteria

Step 7: Calculate relative QoS of each provider w.r.t. user's QoS requirements

This step involves multiplying the user's QoS priorities to the provider's QoS. The results indicate the nearness of each of the provider's to the user's requirements.

Example

$$\begin{bmatrix} 0.5294 & 0.4986 & 0.4705 \\ 0.2941 & 0.1253 & 0.2352 \\ 0.1764 & 0.3760 & 0.2941 \end{bmatrix} \text{X} \begin{pmatrix} 0.5584 \\ 0.3196 \\ 0.1220 \end{pmatrix} = \begin{pmatrix} 0.5123 \\ 0.2329 \\ 0.2545 \end{pmatrix} \begin{matrix} Prov1 \\ Prov2 \\ Prov4 \end{matrix}$$

Step 8: Final selection

The previous steps were based on user's QoS priorities; this step makes the final selection based on the user QoS requirement which is inputted on a scale of low, medium and high. The provider's cost QoS ratings are given below. In the following example input for the QoS rating was low.

Example

$$\begin{array}{l} \text{Prov1 Cost} \\ \text{Prov2 Cost} \\ \text{Prov4 Cost} \end{array} \begin{bmatrix} 3 \\ 5 \\ 8 \end{bmatrix}$$

User's QoS ratings input is low. This is used to calculate the relative cost by the following equation.

Relative Cost rating = Provider's QoS cost * (cost rating value/3)

The cost rating values are: low = 1, medium = 2, high = 3

Relative cost for Prov1 = 3*1/3 = 1

$$\begin{array}{l} Prov1 \\ Prov2 \\ Prov4 \end{array} \begin{bmatrix} 1 \\ 1.6667 \\ 2.6667 \end{bmatrix}$$

Multiply the relative cost to the relative QoS.

$$\begin{bmatrix} 0.5123 * 1 \\ 0.2329 * 1.6667 \\ 0.2545 * 2.6667 \end{bmatrix} = \begin{bmatrix} 0.5123 \\ 0.3881 \\ 0.6786 \end{bmatrix}$$

The results show that provider 4 is best match for the user's requests for low QoS. Provider 1 has the best QoS however due to its higher cost and user requirement for low QoS; provider 4 is selected.

4.4.3. QaComPS Selection Conclusion

This section has described the novel QaComPS selection service. The selection process uses ranking and selection steps to reach the best provider's VM. The efficiency of the selection process is improved by the filtering and ranking as both reduce the number of potential providers. The accuracy of the selection process is mainly concerned with the selection component. The ranking component also plays a role in improving the accuracy as the results from the component are ranked with respect to their nearness to the user's requirements.

4.5. Run Service

The QAComPS Run service is an internal service that is invoked by the user through the QaComPS selection service. Figure 4.12 shows the QAComPS Run service architecture. The service accesses the providers using an API/WSDL interface. The description of the selected provider's VM is passed onto the run service through the selection service. This includes: Provider name (name of the provider); VM size i.e. small, medium, large or very large; Image ID; and start-up and finish time. If there is no image ID the service queries the provider for available VM images.

The main service is managed by the "Service Manager". The first stage is to setup the VM image. This consists of all the application software, OS (Nurmi, Wolski et al. 2009) etc. that is be installed on the VM. VM Images reduce the setup times to a few minutes or even seconds. After acquiring the Image ID the Service Manager creates the new VM and sets it up by installing the VM image. Once the VM setup is complete it is started.

A VM is accessed, via either a static IP or a public DNS key. Static IPs are used in Web browsers while the DNS key is used with a secure shell (SSH) client. Using these keys the users can not only upload and run their code but also restart and shutdown the VM. A job log is maintained for each VM where information such as start time, end time and any errors or exceptions is stored.

A checkpoint service can be used with the Run service to create checkpoints at certain predefined intervals. This is useful for splitting the job into manageable sections which is vital for recovering from service failures. The checkpointing is part of the future work and is not pursued further in this research.

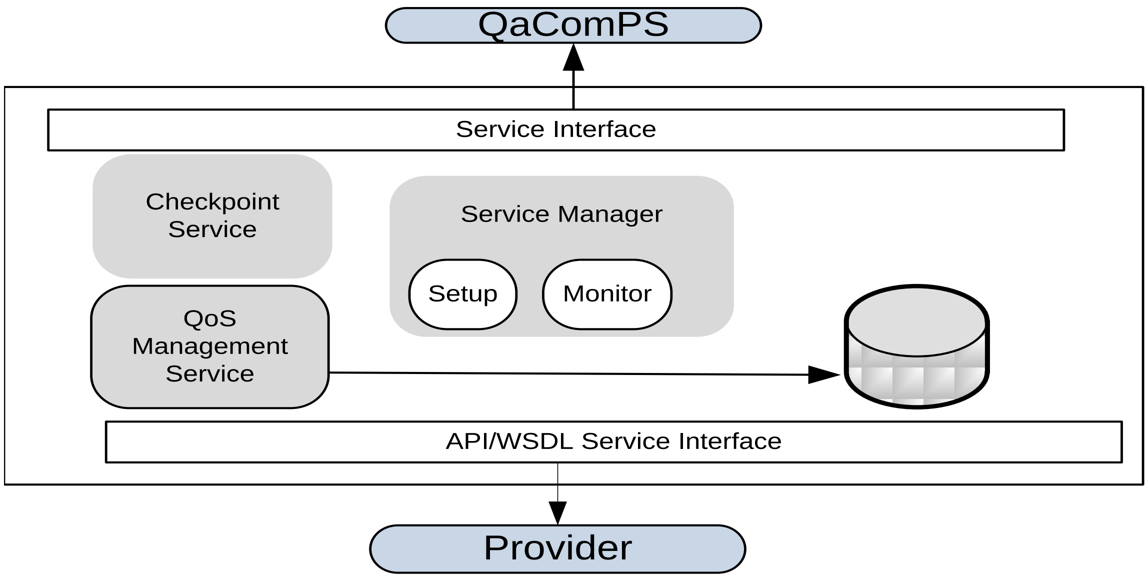


Figure 4.12: Run Service

4.6. Conclusion

This chapter has presented a novel Quality-aware computational Cloud selection (QAComPS) service. This service enabled automatic selection and invocation of Cloud provider's VMs.

The Information Management service is central to the working of the selection service. As it is the only service that has access to the provider's information database. The database is queried for processing each user's request. The Information Management service uses SAWSDL lowering schema mapping to transfer information to the (non semantic) selection service. The SAWSDL lifting schema mapping is used for updating the provider QoS.

The QaComPS selection service potentially saves users money by selecting the cheapest provider's VM. The technologic advance was a novel selection process based on an ontology-based filter, a Euclidean distance based ranker and an AHP based selector. The filter reduced the number of potential providers and improved the efficiency of the selection process. The Euclidean Distance ranked the list of services from which the top three providers were sent to the AHP-based selector. The selection process and returns the best provider’s VM that was then used setup/run by the Run service.

The Run service grants user full access to the selected VM by sharing the public DNS access key or the available static IP (depending on the provider). The access key is used for remotely accessing the VM.

Chapter 5
Implementation of the QaComPS Prototype

This chapter describes the implementation of the QaComPS prototype. The service was implemented in three phases. The first phase was concerned with implementing the Information Management service. Implementation of the QaComPS selection service was undertaken in the second phase; this included implementation of SAWSDL annotations for communication between services. The final phase was to implement the Run service.

The chapter is sub divided into three sections where the first section describes implementation of Information Management service. This is followed in section 5.2 with the implementation of the Selection service. Section 5.3 describes the implementation of the Run service. The implementation of the database is included with the Information Management service while the implementation of the user interface is described with the selection service.

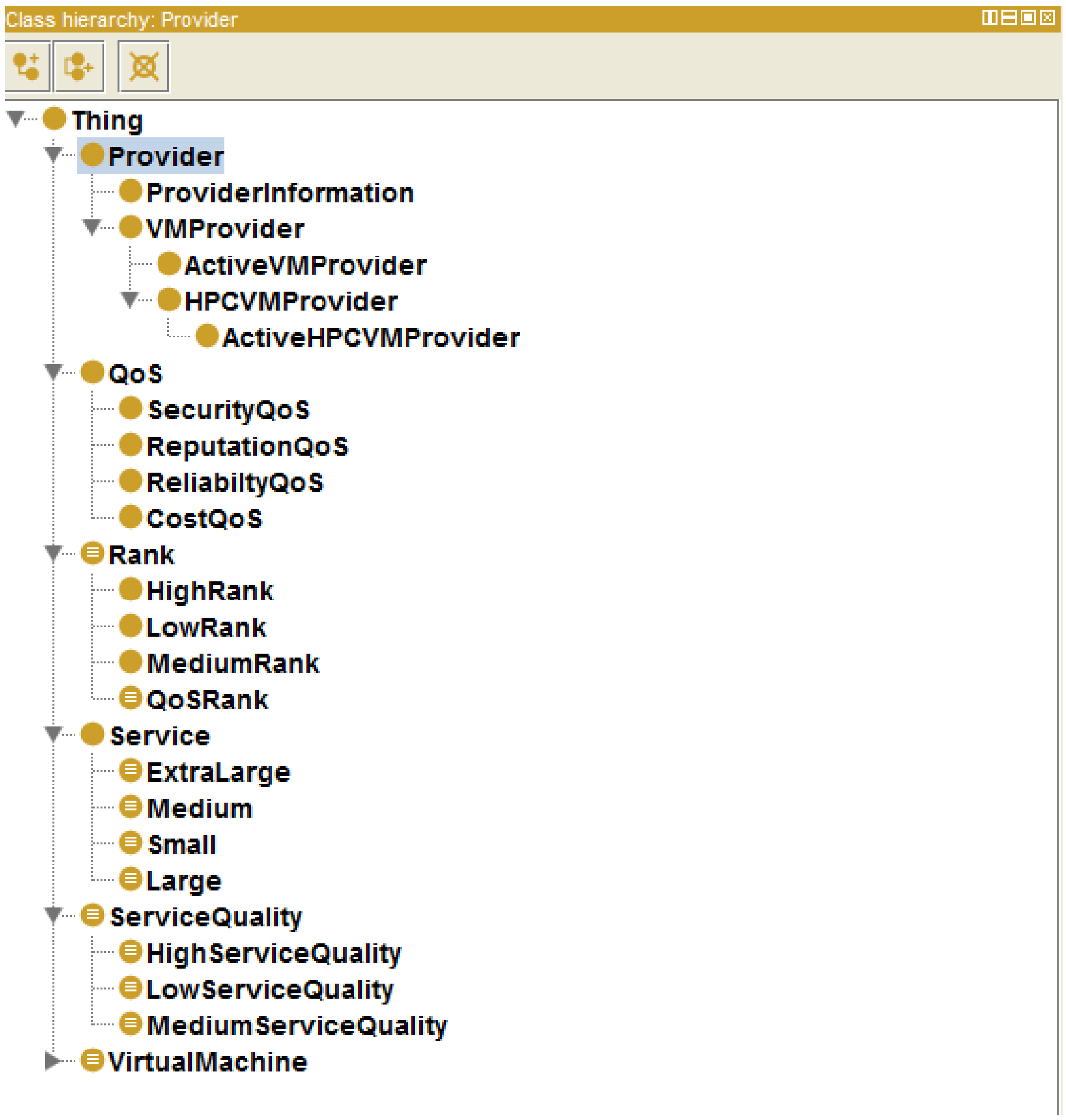
5.1. QaComPS Information Management service Implementation

The QaComPS Information Management service was implemented in three steps. The first step was to implement the service ontology. This was followed by the implementation of RDF manager. The final step was to implement the storage of RDF information in a MySQL database.

5.1.1. Implementing Service Ontology

The QaComPS service ontology was developed using protégé (Somasundaram, Balachandar et al. 2006). Protégé is a free to use ontology editor for modelling knowledge-based applications. The concepts were encoded in OWL as classes, sub-classes and their relationship as properties.

5.1.1.1. Classes



```
Thing
  Provider
    ProviderInformation
    VMProvider
      ActiveVMProvider
      HPCVMProvider
        ActiveHPCVMProvider
  QoS
    SecurityQoS
    ReputationQoS
    ReliabiltyQoS
    CostQoS
  Rank
    HighRank
    LowRank
    MediumRank
    QoSRank
  Service
    ExtraLarge
    Medium
    Small
    Large
  ServiceQuality
    HighServiceQuality
    LowServiceQuality
    MediumServiceQuality
  VirtualMachine
```

Figure 5.1: Service Ontology Classes

The service ontology has six high level classes; these are independent classes. Independent classes can have relations with one another but cannot be parent or subclasses. For example provider and VM classes are related however neither a VM is a type of a provider nor is a provider a type of a VM. Sub classes are type of a parent class for example HighRank is a type of Rank.

The relations between classes are described through object properties. Figure 5.1 shows the high level classes along with their sub classes.

The Virtual Machine class describes a Cloud VM; Provider class describes a Cloud provider while the Rank class describes the QoS rank of the VM. The value for the QoS ranking could either be high, medium or low. The service class describes the size of the VM that is small, medium, large and very large.

5.1.1.2. Properties

The relationships between two concepts (classes) are represented as properties in an ontology. There are two types of properties namely object properties and data properties.

Object properties describe the relationships between classes. For example "Provider hasVirtualMachine VirtualMachine". In this example the provider class is the domain, VirtualMachine is the range and the object property hasVirtualMachine describes the relationship between the domain and the range classes. Some of the object properties from the QaComPS service ontology are given in table 5.1.

Domain	**Object Property**	**Range**
Provider	hasReliability	Reliability
Provider	hasReputation	Reputation
Provider	hasVirtualMachine	VirtualMachine
VM	hasRank	Rank
VM	hasService	Service

Table 5.1: Object Properties in the QaComPS Service Ontology

Data properties describe the type of data associated with a class. For example "Provider hasName Name" where provider is the domain, hasName is the data property and Name is the range. Unlike object properties, in this case "Name" is not a class but a data property of type string, meaning the value for "Name" would be a string. The concept of data properties is the same in programming paradigm where these are referred to as data types and are used for describing the type of data being stored (Motik and Horrocks 2008).

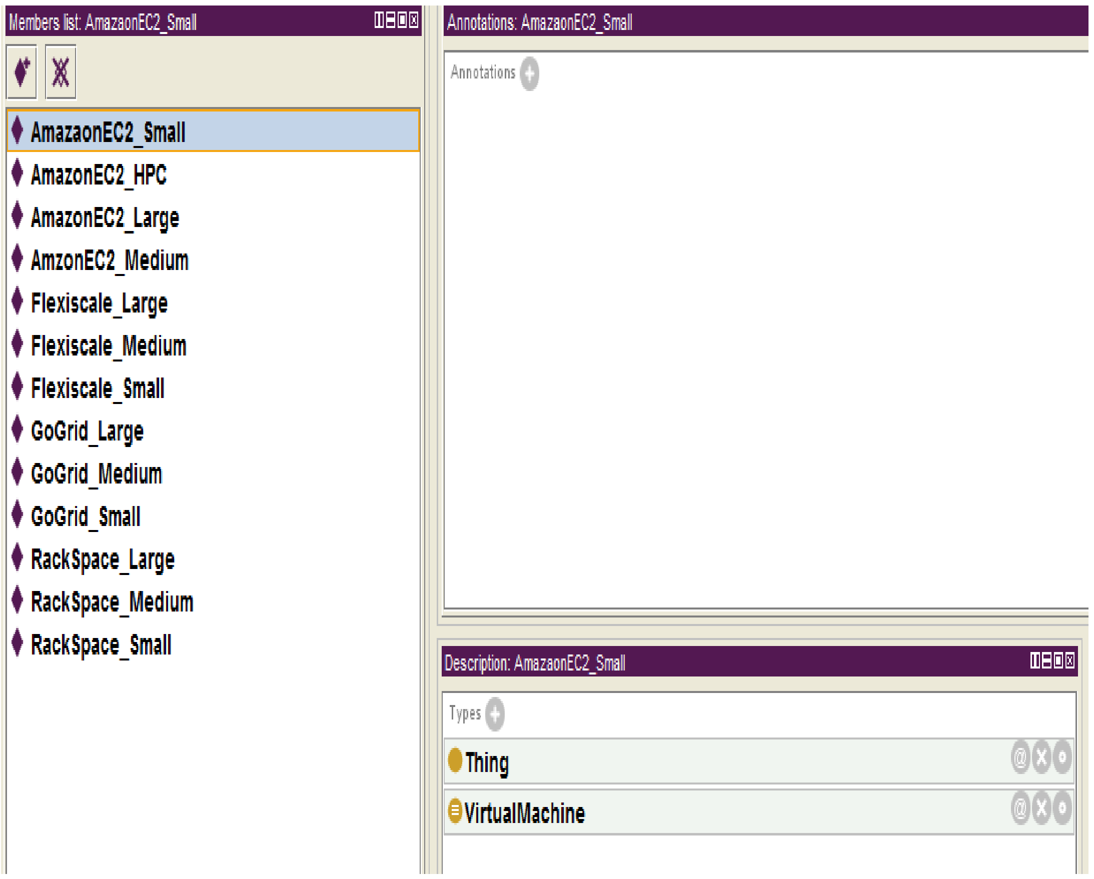
Service ontology data properties are distributed into three groups these are; relating to the Provider, relating to the VM; and relating to the QoS.

Domain	**Data Property**	**Range**
VM	hasCores	Integer
VM	hasMemory	Double
VM	hasNonPersistantStorage	Double
Provider	hasName	String
Provider	hasID	Literal
Provider	hasURL	URI

Table 5.2: Data properties associated with QaComPS Service Ontology

Double is a data type supported by Protégé; it is used to indicate that the value held by this data property will be a number with decimal. Integer and string are used for storing integer numbers and characters strings respectively while literal is used for storing alphanumeric values.

5.1.1.3. Instances


Members list: AmazaonEC2_Small

AmazaonEC2_Small
AmazonEC2_HPC
AmazonEC2_Large
AmzonEC2_Medium
Flexiscale_Large
Flexiscale_Medium
Flexiscale_Small
GoGrid_Large
GoGrid_Medium
GoGrid_Small
RackSpace_Large
RackSpace_Medium
RackSpace_Small

Annotations: AmazaonEC2_Small

Annotations

Description: AmazaonEC2_Small

Types
Thing
VirtualMachine

Figure 5.2: Instances associated with QaComPS Service Ontology

Some of the instances from the QaComPS service ontology are shown in figure 5.2. The figure shows instances for the VirtualMachine class. The naming convention for the instances shows that these belong to four Cloud providers and have four different sizes.

5.1.2. Implementing RDF Manager

The RDF manager component is the central unit in the Information Management service. It is responsible for processing the Cloud provider’s information. The process involves checking conformance with the service ontology, producing RDF for new providers, updating RDF for existing providers and querying the database.

This component was implemented using NetBeans Integrated Development Environment (IDE) (Böck and Tulach 2009) and the Jena API (Jena 2007).

The RDF manager starts by creating a new ontology model. The model contains information regarding the new provider’s entry. Each entry in the new model is described as a Universal Resource Identifier (URI). The newly created model is reasoned against the ontology specification for identifying the inferred model. An inferred model contains the full list of relations such as Provider’s relation to a VM or VM’s relation to the QoS. Even if the relation between provider and QoS was not explicitly mentioned; it would be part of the inferred model.

The code for reasoning owl ontologies and identifying inferred models is available as part of the OWL (Jena) API. The Java/Jena code for invoking the owl reasoner and creating the inferred model is shown in figure 5.3. This code returns the inferred model for the inputted values.

The RDF file for the new provider contains information from the inferred model. Describing each relation explicitly is useful while querying the RDF.

The RDF descriptions generated for the Provider 1’s small VM are shown in figure 5.4. These consist of the QoS information for the provider but do not include the rules and object properties. As the original file was too long see appendix 2.

```
public Model modelReasoner()

{

   Reasoner reasoner1 = ReasonerRegistry.getOWLReasoner();

   reasoner1 = reasoner1.bindSchema(newModel);

   Model inferredModel = ModelFactory.createInfModel(reasoner1, newModel);

   return inferredModel;}
```

Figure 5.3: Code for modelReasoner

```
<rdf:RDF

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

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

   xmlns:xsd="http://www.w3.org/2001/XMLSchema#"

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

   xmlns:j.0="http://localhost/Test/ServiceOntology.owl#" >

  <rdf:Description
rdf:about="http://localhost/Test/ServiceOntology.owl#VirtualMachine">

   <j.0:hasUpload>30</j.0:hasUpload>

   <j.0:hasDownload>50</j.0:hasDownload>

   <j.0:hasCores>1</j.0:hasCores>

   <j.0:hasNonPersistentStorage>160</j.0:hasNonPersistentStorage>

         <j.0:hasMemory>1.7</j.0:hasMemory>

         <j.0:hasCost>0.8</j.0:hasCost>

   <j.0:hasType>small</j.0:hasType>

  </rdf:Description>

  <rdf:Description
rdf:about="http://localhost/Test/ServiceOntology.owl#ProviderInformation">

   <j.0:hasWSDL rdf:resource="http://localhost/Test/Provider1.wsdl#WSDL"/>

   <j.0:hasWebURL>http://http://Provider1.com</j.0:hasWebURL>

   <j.0:hasName>Provider1</j.0:hasName>
```

Figure 5.4: RDF description for Provider 1’s small VM

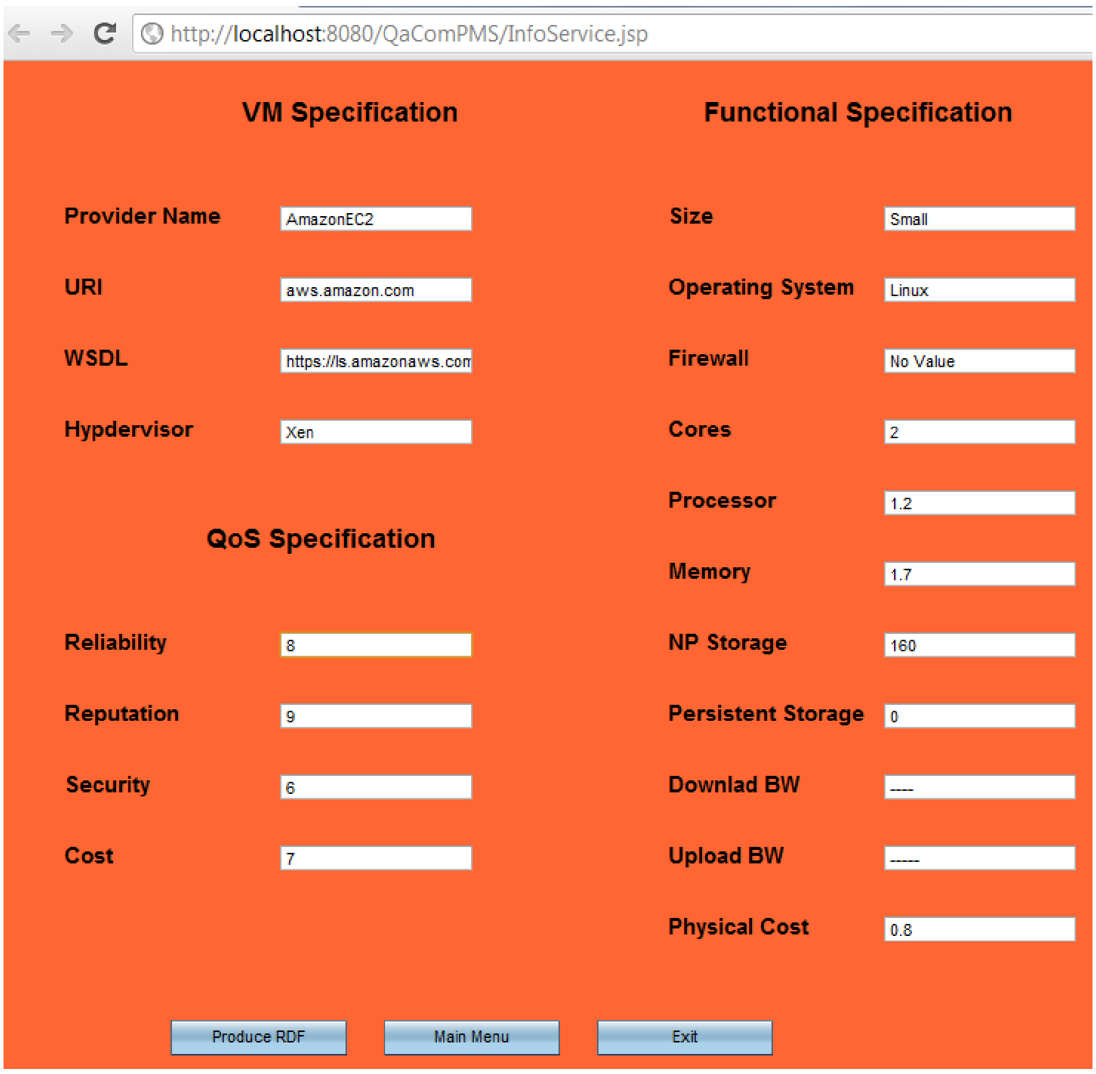
5.1.3. Implementation of the Database

There were two options for storing RDF files in a database; a semantic database such as RDFDB (R.V.Guha 2000) or a relational database management systems (RDBMS) such as MySQL (MySQL 2005). The advantage of using semantic database was less processing as the RDF can be queried directly using SPARQL (Prud'Hommeaux and Seaborne 2008). These databases are in early development and not as efficient or secure as the RDBMS. In favour of RDBMS: MySQL, Oracle have been in continuous development for years and are highly trusted RDBMS (MySQL 2005). The disadvantage of using RDBMS is that additional processing was required. The RDF files are stored as data objects in the RDBMS. These would be extracted using Structured Query Language (SQL) query after which they would be queried using SPARQL.

The Information Management service stores information in a MySQL RDBMS. The RDF information of each provider is stored as a Character Large Object (CLOB) entry in the database. CLOB is used for storing RDF data as character object. If a semantic database were to be used the RDF information would be queried directly using SPARQL.

5.1.4. Information Management service User Interface

The interface for the QaComPS Information Management service is shown in figure 5.5. It was developed using the NetBeans Java IDE (Böck and Tulach 2009).



http://localhost:8080/QaComPMS/InfoService.jsp

VM Specification

Provider Name	AmazonEC2
URI	aws.amazon.com
WSDL	https://ls.amazonaws.com
Hypdervisor	Xen

QoS Specification

Reliability	8
Reputation	9
Security	6
Cost	7

Functional Specification

Size	Small
Operating System	Linux
Firewall	No Value
Cores	2
Processor	1.2
Memory	1.7
NP Storage	160
Persistent Storage	0
Downlad BW	----
Upload BW	-----
Physical Cost	0.8

Figure 5.5: Information Management service Interface

5.1.5. Implementing Filter Component

The SQL and SPARQL queries for the filtering are described in chapter 4; see section 4.2.2.1.

Figure 5.6 contains the code snippet for filtering the providers. The queryString represents the SPARQL query. A query is created dynamically for each request. The query is executed using the inferred ontology model. The results of the query are formatted and then passed onto the ranking component.

```
Query query = QueryFactory.create(queryString);

QueryExecution qe = QueryExecutionFactory.create(query, model);

ResultSet results = qe.execSelect();

OutputStream o = new FileOutputStream(new File("C:/QComBroker/results/QueryResults.txt"));

ResultSetFormatter.out(o, results, query);
```

Figure 5.6: Code for filtering providers

5.2. Implementation of QaComPS Selection Service

The selection service implementation consists of implementing the ranking and selection components. The QaComPS selection service was also implemented using Java with the NetBeans IDE. The service requires a minimalistic set of inputs from the user. These are for the VM size, QoS and QoS weights.

5.2.1. Implementing Ranking Component

The implementation for the ranking component is given in the following steps which include code snippets. The code snippets used in these steps are for the key steps of the ranking process. Including the entire code was not feasible due to its large size upto 10,000 lines of code.

Step 1: Create Comparison Matrix

The first step is to transform the user inputs into a comparison matrix. The code snippet shown in figure 5.7 convert user inputs (identified as comparisons [ ]) to a two the comparision Matrix. This involves populating the diagonal with ones.

This is followed by storing the inputted values on the right hand side and the reciprocated values in the left hand side of the matrix.

```
for(int i = 0; i < criteriaNumber; i++)
        {comparisionMatrixData[i][i] = 1;}
        for(int i = 0; i < criteriaNumber; i++)
        { y++;
                for(int j = y; j < criteriaNumber; j++)
                {comparisionMatrixData[i][j] = comparisons[x];
                comparisionMatrixData[j][i] = 1/comparisons[x];
        x++;
        }}
```

Figure 5.7: Code for transforming inputs to matrix

Step 2: Calculate criteria weights

This requires the criteria sum (sum of values in each column). The sum step has been skipped as it simple addition. The weights are calculated by dividing each with the sum of its column.

```
for (int i = 0; i < criteriaNumber; i++)
{       val = 0.0;
        for (int j = 0; j < criteriaNumber; j++)
        {val += comparisonMatrix[i][j] / riteriaWeightComponentArray[j];}
        criteriaWeightArray[i] = (1.0 / criteriaNumber) * val;
}
```

Figure 5.8: Code for calculating criteria weights

Step 3: Calculate denominator for normalizing equation

```
for (int i = 0; i < criteriaNumber; i++)

{ val = 0.0;

        for (int j = 0; j < providerNumber; j++)

        { val += Math.pow(historicDataMatrix[i][j], 2.0); }

        normalisedPerformanceComponentArray[i] = Math.sqrt(val);

}
```

Figure 5.9: Code for calculating denominator

Step 4: Calculate the performance and weighted normalized matrix

The normalized performance matrix is calculated by dividing providers QoS data with the normalized performance matrix from the previous step.

The normalized performance matrix is multiplied with the QoS weights to calculate the weighted normalized performance matrix.

Step 5: Relative Euclidean Distance

The code snippet for calculating the relative Euclidean distance is shown in figure 5.10. Where Math.pow is used for squaring the values and Math.sqrt is for calculating the square root.

The euclideanDistanceArray contains the relative Euclidean distances. It is sorted in ascending order to get the ranked results.

```
for (int j = 0; j < providerNumber; j++)
{double val = 0.0;
  for (int i = 0; i < criteriaNumber; i++)
  { val += Math.pow(historicDataMatrix[i][j], 2.0); }
   euclideanDistanceComponentArray[j] = Math.sqrt(val);}
double[] euclideanDistanceArray = new double[providerNumber];
  for (int j = 0; j < providerNumber; j++)
  {double val = 0.0;
    for (int i = 0; i < criteriaNumber; i++)
     {val +=  Math.pow(weightedNormalisedPerformanceMatrix[i][j]
            - criteriaWeightArray[i] * requirementsArray[i]/
            euclideanDistanceComponentArray[j],2.0); }
  euclideanDistanceArray[j] = Math.sqrt(val);
}
```

Figure 5.10: Code for calculating relative Euclidean distance

5.2.2. Implementing Selection Component

This section consists of snippets of Java code used for implementing the selection component. The code snippet shown in figure 5.10 is for squaring a 3 dimensional matrix.

```
x[0] = (a1*a1) + (a2*b1) + (a3*c1);

x[1] = (a1*a2) + (a2*b2) + (a3*c2);

x[2] = (a1*a3) + (a2*b3) + (a3*c3);


x[3] = (b1*a1) + (b2*b1) + (b3*c1);

x[4] = (b1*a2) + (b2*b2) + (b3*c2);

x[5] = (b1*a3) + (b2*b3) + (b3*c3);


x[6] = (c1*a1) + (c2*b1) + (c3*c1);

x[7] = (c1*a2) + (c2*b2) + (c3*c2);

x[8] = (c1*a3) + (c2*b3) + (c3*c3);
```

Figure 5.11: Code for squaring a 3d matrix

Figure 5.12 shows the code snippet for calculating the eigenvector for a three dimensional matrix. The comments inside the code are represented by the // sign. The difference between the two Eigenvectors is less than 0.001. Square3d represents the function square 3d whose code is shown in figure 5.11. eigAdd3d function returns a normalized array buy dividing each element with the corresponding row sum.

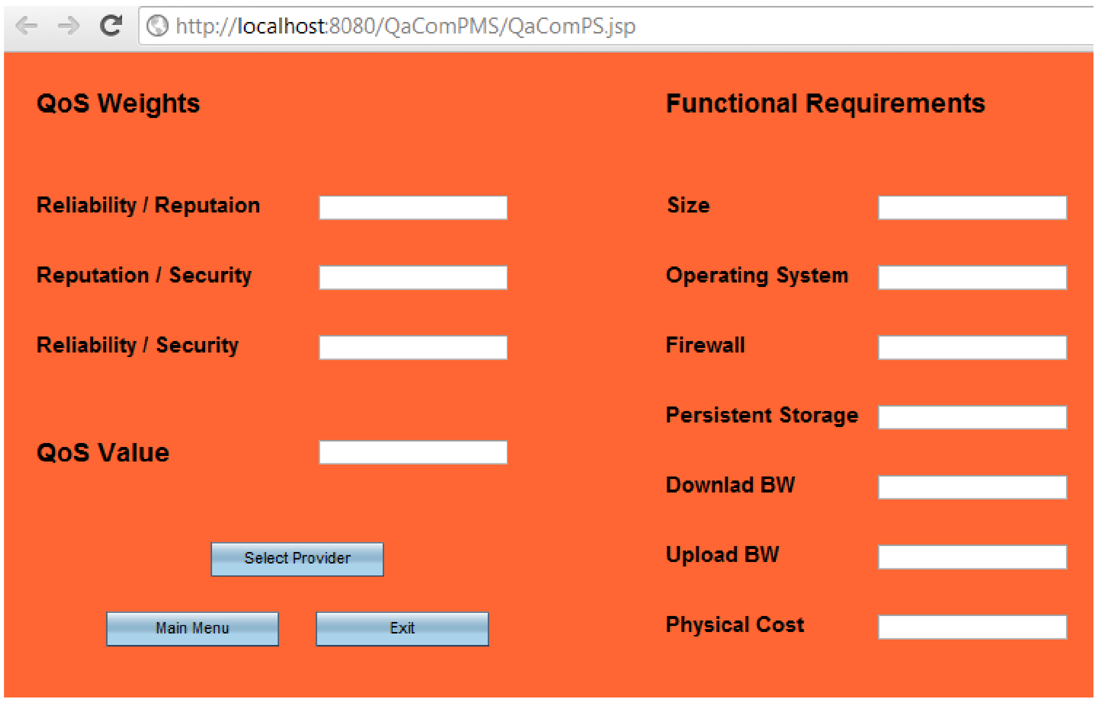
```
while(true)

{

   //step3 calc first eig

   relE1 = eigAdd3D(rel);

   //step4 sqr agian

   rel = square3D(rel[0],rel[1],rel[2],rel[3],rel[4],rel[5],rel[6],rel[7],rel[8]);

   //step5 calc second eig

   relE2 = eigAdd3D(rel);

   if(relE1[3]-relE2[3]>-0.001 && relE1[3]-relE2[3]<0.001

         &&

     relE1[4]-relE2[4]>-0.001 && relE1[4]-relE2[4]<0.001

         &&

     relE1[5]-relE2[5]>-0.001 && relE1[5]-relE2[5]<0.001)

   {    break;    }

 }
```

Figure 5.12: Code for calculating Eigenvector for a 3d matrix

5.2.3. QaComPS User Interface

The user interface has QoS inputs for weights which are inputted with reference to QoS parameters. For example if the reliability is twice as important are reputation the first input would be 1,2.

The functional inputs values for size and operating system are compulsory while the others are optional. Functional input for physical cost represent amount of money the user intends to spend.



http://localhost:8080/QaComPMS/QaComPS.jsp

QoS Weights

Reliability / Reputaion	
Reputation / Security	
Reliability / Security	
QoS Value	

Functional Requirements

Size	
Operating System	
Firewall	
Persistent Storage	
Downlad BW	
Upload BW	
Physical Cost	

Figure 5.13: QaComPS User Interface

5.3. Implementing RUN Service

The Run service is implemented explicitly for each provider; as each provider has its own access mechanism. The following section describes the example of accessing Amazon EC2.

5.3.1. Running EC2 VMs

Amazon EC2 offers a wide array of APIs for remotely accessing Amazon EC2 VMs. In case of this work Amazons Java APIs was used; this included an EC2_Wrapper class which had the following predefined methods:

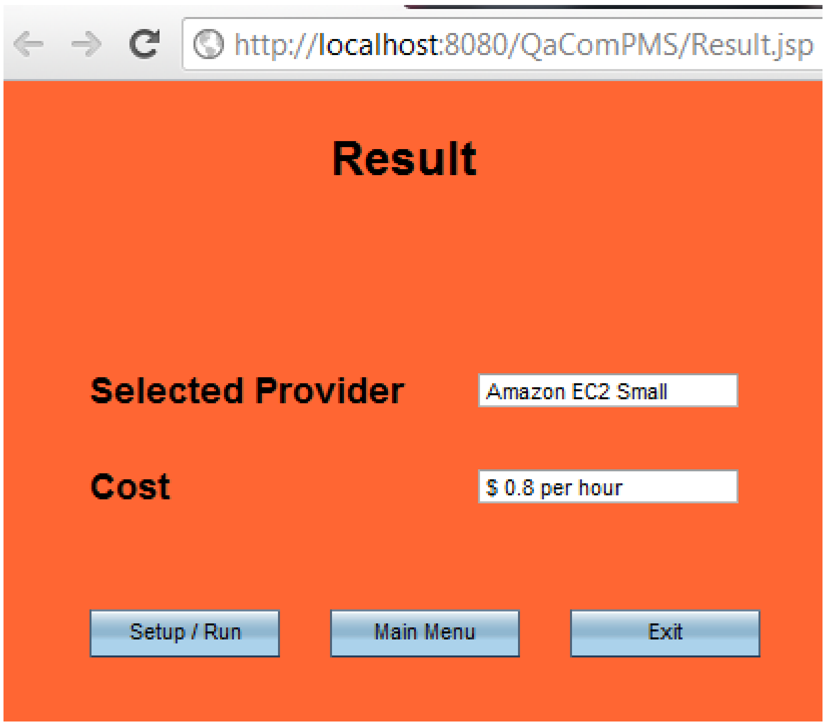
Creating EC2 VMs: createAMInstances (AMIid, min, max, keyPairName, instanceType, availabilityZone)

AM stands for Amazon Machine; min and max represent the minimum and maximum number of VMs that need to be created; keyPairName represents unique security key; instanceType represents VM size; availabilityZone is the geographical zone where the VM would be located.

Destroy VMs: terminateAMIs(); this methods will terminate the current VM

5.3.2. Run Service Interface

Figure 5.14 represents the two steps that form the run service. The first step requires the user to make the final selection by clicking on the run service. At this point the user can quit by pressing main menu or exit if he presses the Setup/Run button the selected VM would be initiated and the user would be given the public DNS or the static IP.



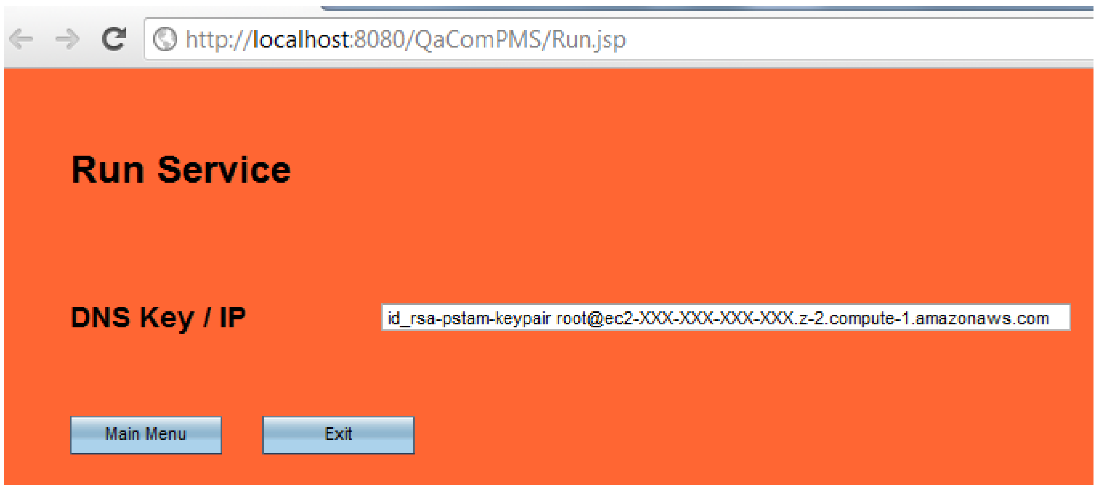
http://localhost:8080/QaComPMS/Result.jsp

Result

Selected Provider: Amazon EC2 Small

Cost: $ 0.8 per hour

Figure 5.14a: Make final selection


http://localhost:8080/QaComPMS/Run.jsp

Run Service

DNS Key / IP: id_rsa-pstam-keypair root@ec2-XXX-XXX-XXX-XXX.z-2.compute-1.amazonaws.com

Figure 5.14b: Run selected Service

5.4. Conclusion

This chapter has presented the implementation of the QaComPS service. The service could have been implemented in a number of programming languages with Java being the preferred option due to its supporting APIs for ontologies and RDF.

The Information Management service was implemented using protégé which was effective for not only describing ontologies but for verifying onotlogies using built-in tools. The implementation for the QaComPS service was over ten thousand lines of code; key parts of this code were described in the chapter. The implementation of the run service was the most difficult part as each provider had it own set of requirements and access methods which meant implementing each provider explicitly.

Chapter 6
Evaluation

The previous chapter described the implementation of the QaComPS. This chapter presents the quantitative and qualitative evaluation for the QaComPS. The evaluation process was to assess; how far QAComPS system achieved the aim of the research; i.e. “select the best (cheapest) provider’s VM while fulfilling all the QoS constraints”. Recording the positives and negatives associated with the use of QaComPS were also part of the evaluation. The QaComPS was evaluated for accuracy and efficiency where accuracy was the measure of nearness to the ideal provider’s service while efficiency was the measure of the response time for user’s queries.

It can be recalled that the QaComPS consisted of three sub services; Information Management service, Selection service and the Run service (see section 4.1). The Information Management service was evaluated implicitly as part of the selection service. The QaComPS selection service was evaluated quantitatively for accuracy and efficiency of the selection process. The selection service was evaluated against AHP and QMP selection approaches (see section 2.2.2). QaComPS was evaluated qualitatively by computational modellers who were interviewed earlier during requirements analysis. The evaluation experiments were carried out on a machine with 4GB of memory, a 2.5 GHz dual core processor and 250GB storage.

The chapter is divided into four sections where section 6.1 describes the quantitative evaluation of QaComPS using simulated providers. The second section describes the evaluation of QaComPS against the AHP and QMP. Section 6.3 describes the quantitative evaluation experiments using actual providers. The final section contains the qualitative evaluation for the QaComPS.

6.1. Quantitative Evaluation of QaComPS

6.1.1. Objective

The objective of this experiment was to evaluate the effectiveness of the QaComPS selection process.

6.1.2. Setup

Ten simulated IaaS Cloud providers were created. The decision to use just ten providers was based on the effort it took to produce and process each simulated provider. Each provider offered four VM sizes these are Small, Medium, Large and Very Large.

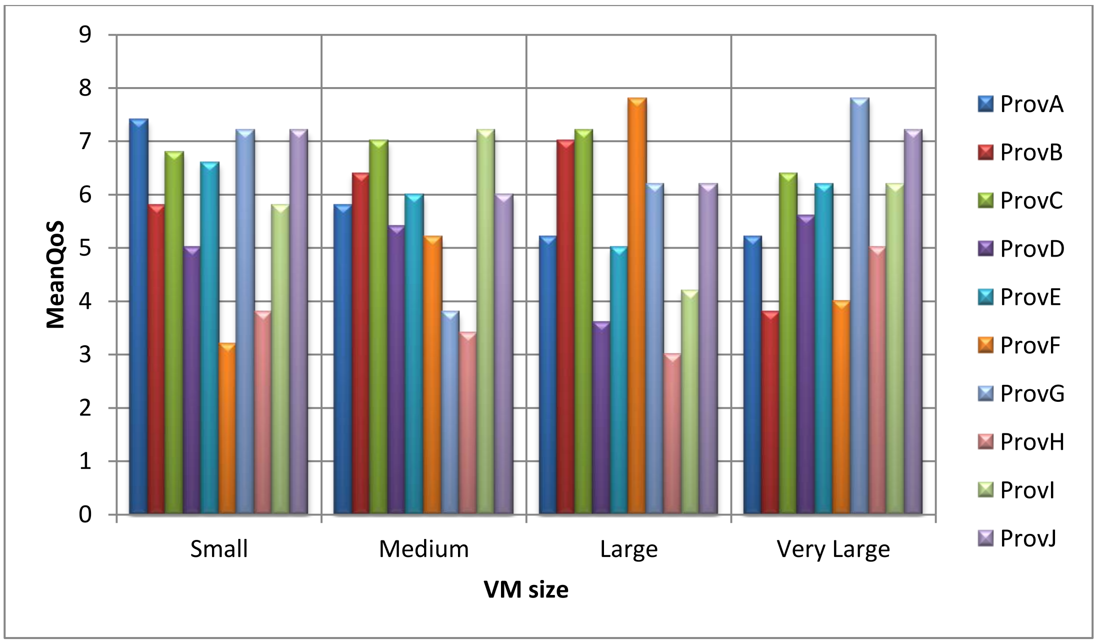


Figure 6.1: Mean QoS for the Simulated Providers

Figure 6.1 shows the ten simulated providers named alphabetically from ProvA to ProvJ. The horizontal line of the graph represents the size of the VM which is distributed into four parts Small, Medium, Large and Very Large. The vertical line on the graph represents the MeanQoS which is calculated using the MeanQoS equation (see 4.3.1). The representation of provider QoS as bars instead of points is due to the way the selection algorithm operates. For example request for low QoS would include results for medium and high while for high QoS would return results for only high QoS. Table 6.1 contains the randomly generated individual values for each provider's QoS parameters.

Name	**Cost**	**Reliability**	**Reputation**	**Security**
ProvASmall	8.0	7.0	9.0	5.0
ProvBSmall	8.0	3.0	7.0	3.0
ProvCSmall	7.0	9.0	2.0	9.0
ProvDSmall	7.0	5.0	1.0	5.0
ProvESmall	7.0	9.0	1.0	9.0
ProvFSmall	4.0	4.0	1.0	3.0
ProvGSmall	8.0	9.0	4.0	7.0
ProvHSmall	4.0	4.0	3.0	4.0
ProvISmall	7.0	7.0	4.0	4.0
ProvJSmall	7.0	8.0	8.0	6.0
ProvAMed	8.0	9.0	1.0	3.0
ProvBMed	9.0	2.0	8.0	4.0
ProvCMed	7.0	9.0	8.0	4.0
ProvDMed	6.0	7.0	7.0	1.0
ProvEMed	4.0	9.0	5.0	8.0
ProvFMed	5.0	8.0	1.0	7.0
ProvGMed	4.0	4.0	4.0	3.0
ProvHMed	4.0	3.0	3.0	3.0
ProvIMed	9.0	4.0	6.0	8.0
ProvJMed	6.0	9.0	5.0	4.0
ProvALarge	6.0	4.0	1.0	9.0
ProvBLarge	9.0	4.0	8.0	5.0
ProvCLarge	8.0	4.0	8.0	8.0

ProvDLarge	5.0	3.0	2.0	3.0
ProvELarge	4.0	6.0	6.0	5.0
ProvFLarge	8.0	9.0	7.0	7.0
ProvGLarge	4.0	6.0	9.0	8.0
ProvHLarge	4.0	2.0	4.0	1.0
ProvILarge	4.0	5.0	5.0	3.0
ProvJLarge	7.0	4.0	7.0	6.0
ProvAvLarge	8.0	1.0	8.0	1.0
ProvBvLarge	4.0	1.0	1.0	9.0
ProvCvLarge	6.0	7.0	7.0	6.0
ProvDvLarge	5.0	7.0	3.0	8.0
ProvEvLarge	6.0	7.0	4.0	8.0
ProvFvLarge	5.0	2.0	2.0	6.0
ProvGvLarge	8.0	9.0	7.0	7.0
ProvHvLarge	4.0	8.0	3.0	6.0
ProvIvLarge	9.0	5.0	1.0	7.0
ProvJvLarge	7.0	8.0	5.0	9.0

Table 6.1: List of simulated Cloud Providers

The next requirement for the experiment was a set of controlled user requests these are shown in table 6.2. The user requests only vary one field at a time while keeping the others constant; in order to analyze the impact of change.

It can be observed from table 6.1 that the value of cost is comparatively higher than the other parameters. The entries in table 6.1 were generated through a random number generator. This returned number between 0 and 1 which were multiplied by nine to produce numbers between one and nine. These numbers had a decimal point which was rounded up or rounded off to convert the numbers to integers. In case of cost the values were rounded up as high cost QoS meant cheaper provider. For example a random value of 3.1 for cost would become 4.0. In case of other QoS properties the number were rounded for example 3.1 would became 3 while 3.8 became 4.

Request ID	**Reliability**	**Reputation**	**Security**	**Cost**
1	1	1	1	9
2	2	1	1	9
3	3	1	1	9
4	4	1	1	9
5	5	1	1	9
6	6	1	1	9
7	7	1	1	9
8	8	1	1	9
9	9	1	1	9
10	1	2	1	9
11	1	3	1	9
12	1	4	1	9
13	1	5	1	9
14	1	6	1	9
15	1	7	1	9
16	1	8	1	9
17	1	9	1	9
18	1	1	2	9
19	1	1	3	9
20	1	1	4	9
21	1	1	5	9
22	1	1	6	9
23	1	1	7	9
24	1	1	8	9
25	1	1	9	9

Table 6.2: User requests

The QoS parameters were varied within the range of one and nine therefore each parameter can be varied eight times. For the first request only cost QoS is considered therefore the other three are fixed at the lowest rating. This would return the overall cheapest provider. User requests two to nine were for the reliability with the reputation and security constant at the lowest. The next eight requests kept the value of reliability and security at one and varied the reputation between two and nine. The last eight requests kept the values of reliability and reputation constant at lowest while changing the values of security from two to nine.

6.1.3. Results

The results shown in figure 6.2 can be explained by looking at table 6.1 which contains the provider information. Request one is interesting as no provider had a rating of nine for cost while three providers had a rating of eight out of these ProvASmall has the best Mean QoS therefore it was selected. It is evident that only two providers had the highest reliability rating ProvCSmall and ProvGSmall. ProvGSmall offers a cheaper cost therefore it was selected over ProvCSmall. For reputation requests only ProvASmall had a reputation rating of nine therefore it was selected for all reputation ratings. For security ratings ProvCSmall and ProvESmall both had a rating of nine. In this case this MeanQoS was compared where ProvCSmall is ahead of ProvESmall.

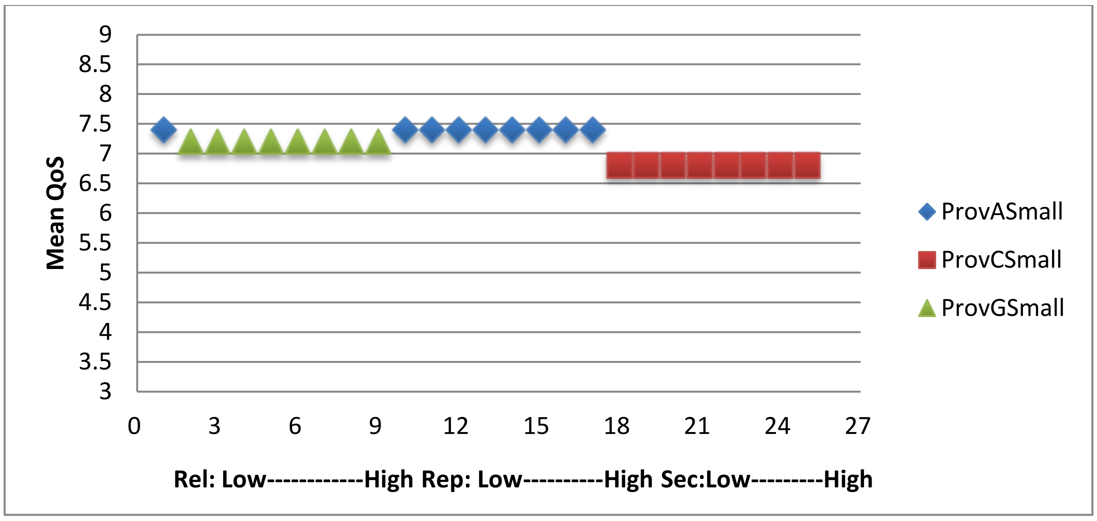


Figure 6.2 presents the simulation results for small VM.

The reason that only three providers got selected is the small number of QoS parameters. As only three QoS parameters were varied while cost remained constant.

The queries for medium VM returned ProvAMed, ProvBMed, and ProvIMed as shown in figure 6.3. ProvBMed and ProvIMed both had a cost rating of nine which meant that they were the cheapest. The selection was made using MeanQoS where ProvIMed had the higher rating. The queries for low reliability returned ProvIMed as it had the highest cost rating with low reliability. ProvAMed is selected for medium reliability request. It had the highest reliability rating with a high cost rating. ProvAMed was also selected for high reliability requests as it had the highest reliability with high cost. ProvBMed is selected for all reputation requests. The selection is based on ProvBMed's cost rating of nine and reputation rating of eight; both of which are high and unmatched. For all security requests ProvIMed was selected which had a cost rating of nine with a security rating of eight.

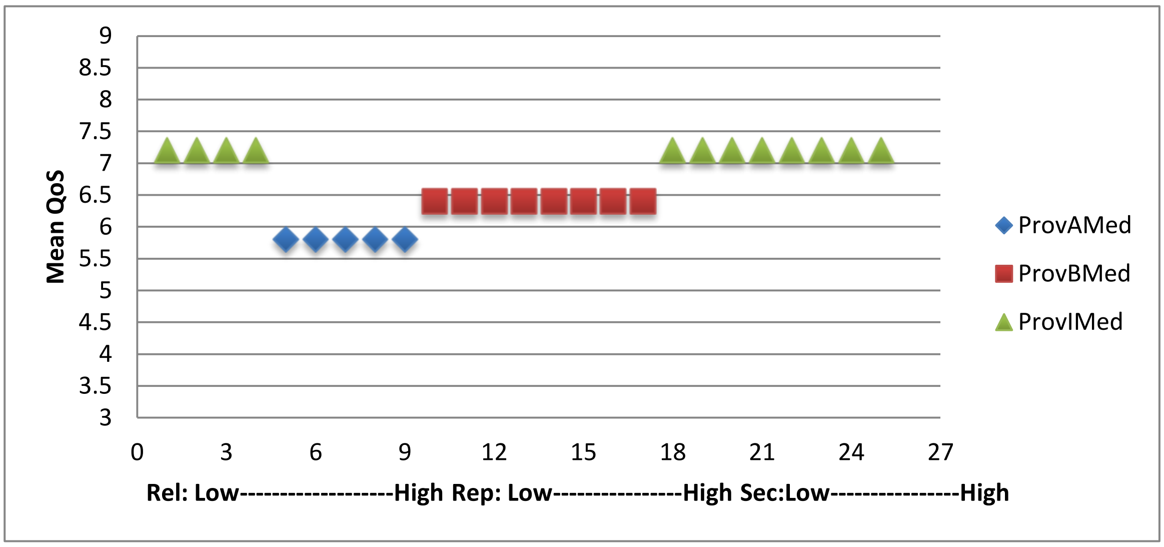


Figure 6.3: Simulation results for Medium VM

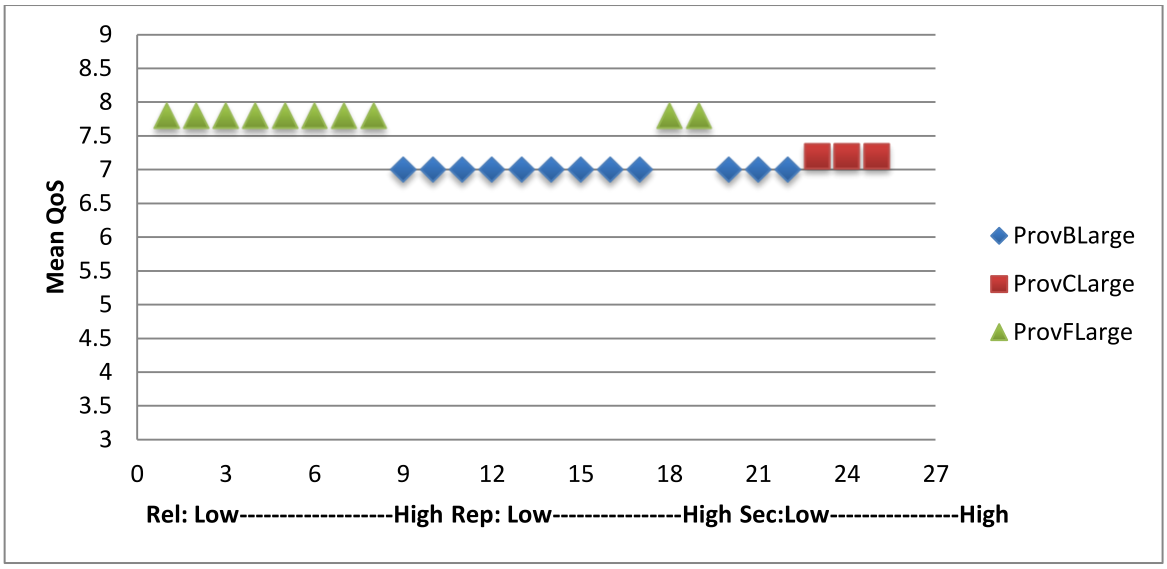


Figure 6.4: Simulation results for Large VM

The simulation results for Large VM are shown in figure 6.4. ProvFLarge had a high cost rating with the highest Mean QoS. Therefore it was selected for the first request. The reliability rating for ProvFLarge was nine therefore it was also selected for the reliability requests. ProvBLarge was selected for the reputation requests as it had an unmatched reputation rating of eight with a cost rating of nine.

The results for security requests were interesting as for each low, medium and high security requests a different provider was selected. This was due to the lack of a dominant provider as was the case for medium VM reputation and security requests. The first two requests for low security returned ProvFLarge as it had the highest cost rating and a low security rating. The medium security requests returned ProvBLarge which had a medium security rating with a high cost rating. ProvCLarge had the best security rating with a medium cost rating therefore it was selected only for high security requests.

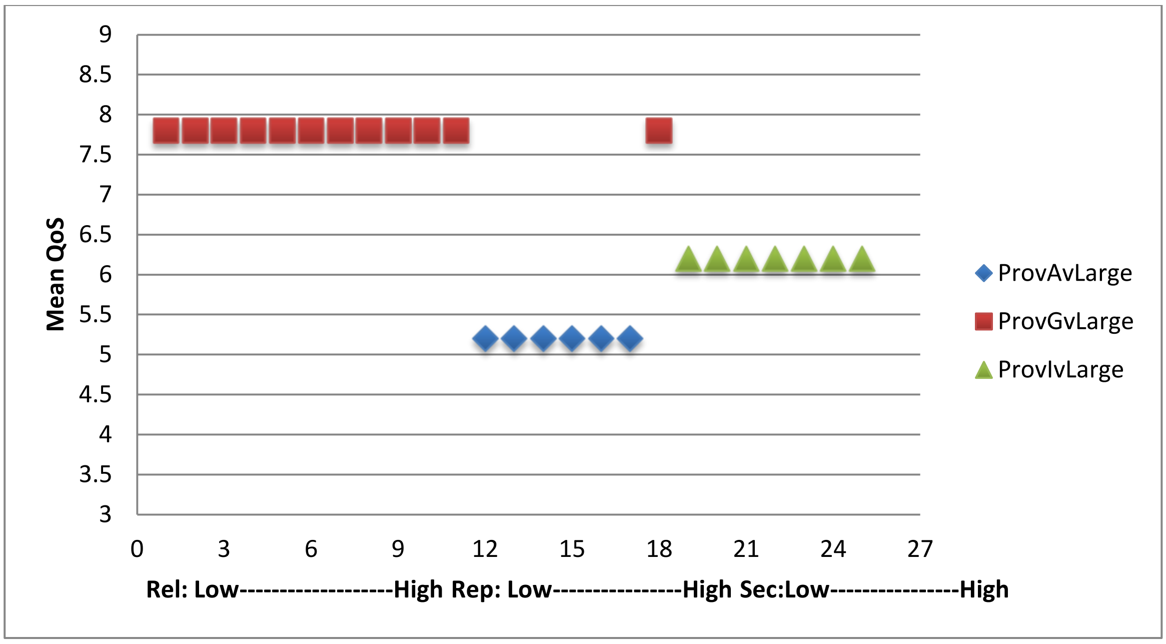


Figure 6.5: Simulation results for Very Large VM

The results for Very Large VM are given in Graph 5. ProvGvLarge had the best MeanQoS rating along with a cost rating of eight therefore it was selected for the first request. ProvGvLarge was also selected for the reliability requests as it had a rating of nine for reliability. For low reputation requests ProvGvLarge was selected due to its low reputation rating with a high cost rating. For medium and high reputation requests ProvAvLarge was selected which had a high rating of eight for reputation and the same for cost. For low security requests ProvGvLarge was selected which had a security rating of three with high cost rating. For the rest of the requests ProvIvLarge was selected due to its cost rating of nine with a security rating of eight.

6.1.4. Analysis of Results

As the input requests were controlled the results were calculated beforehand. These were compared to the above results and there were no discrepancies; both sets had the same results. The objective of the experiment was fulfilled as QaComPS did select the cheapest provider while matching user specified QoS. Due to the random generation of provider QoS some providers dominated as in the case of small and very large VMs. In other cases the selection returned separate providers for each set of requests such as the case of security request for large VM.

6.2. Experiment 2: Comparing QaComPS against QMP and AHP

QaComPS prototype was implemented as a service therefore in order to undertake the evaluation experiment; AHP and QMP services were developed and tested. The AHP service was implemented using (Haas and Meixner 2005) while the QMP service was implemented using (ELEYAN, Amna et al. 2004).

6.2.1. Objective

The objective of this experiment was to identify whether QaComPS can outperform QMP and AHP selection processes in terms of accuracy.

6.2.2. Setup

Twenty five simulated providers were created; each offered a different set of QoS metrics while all of them offered the same computational resources. The simulated providers offered small, medium, large and very large VMs. Resource information associated with small, medium, large and very large came from public Cloud providers as shown in Table 6.1. Twenty five user requests were created with eight user requests for each low, medium and high QoS as shown in table 6.2. These were controlled requests whose output was calculated beforehand. Each user request was passed to AHP, QMP and QAComPS.

6.2.3. Results

Figure 6.6, 6.7, and 6.8 show the results with the horizontal axis showing the user request while the vertical axis showing the cost effect marked on a scale of one to nine. The cost effectiveness is the measure of the nearness to the cheapest provider meeting the QoS constraints. A value of nine for cost effect indicates that the selected provider met all the QoS criteria and was the cheapest while a value of 1 would mean that none of the QoS criteria were met. Values of five and higher indicate that QoS criteria excluding cost were met.

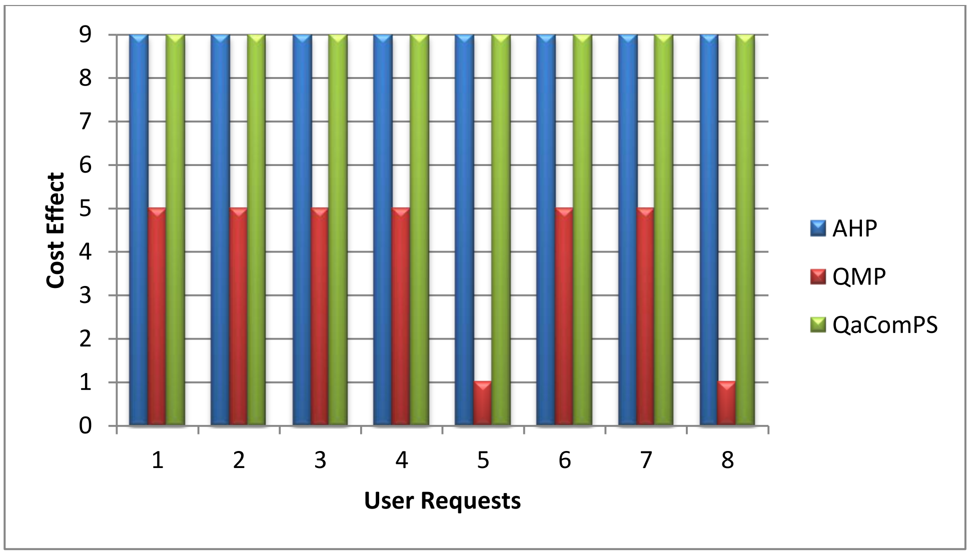


Figure 6.6 MatchMaker Comparisons (High QoS)

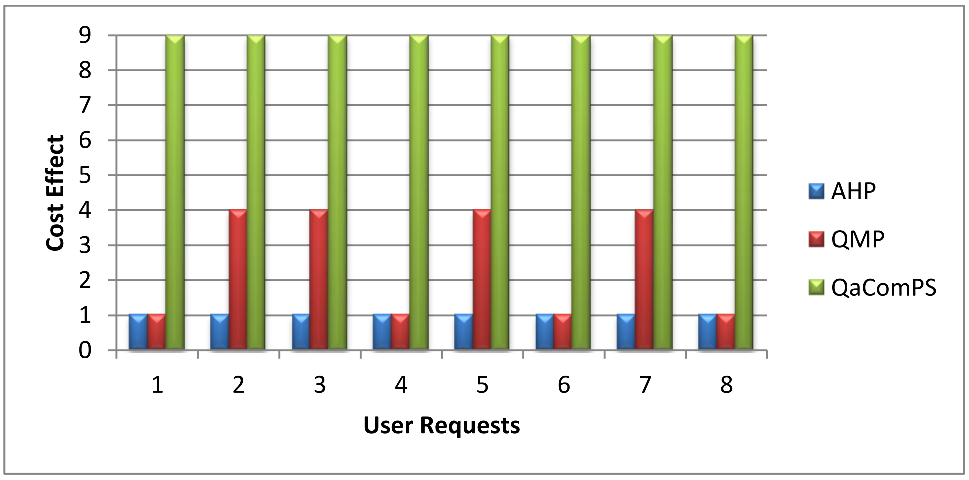


Figure 6.7 MatchMaker Comparisons (Medium QoS)

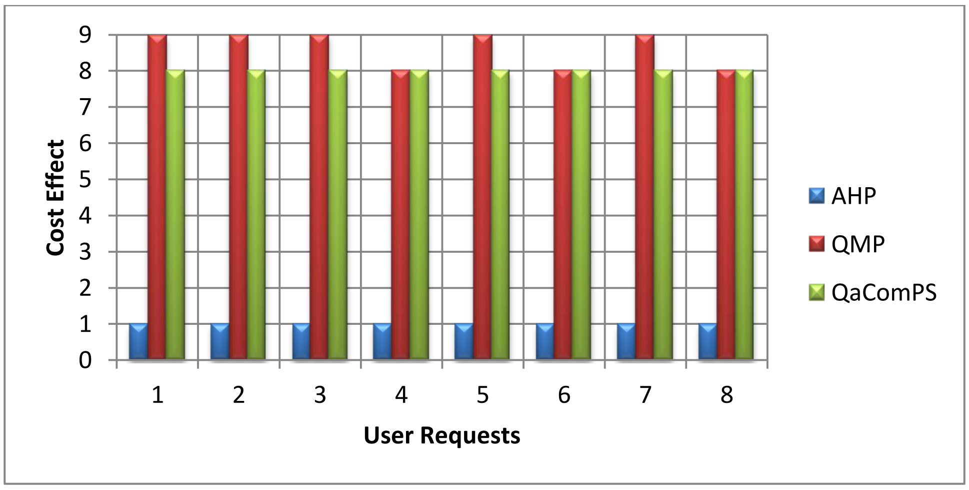


Figure 6.8 MatchMaker Comparisons (Low QoS)

Figure 6.6 shows the results for high QoS. It can be observed in figure 6.6 that AHP and QaComPS were effective for high QoS requests while QMP was not so effective. Figure 6.7 shows the results for medium QoS where QaComPS was the most effective. Figure 6.8 shows the results for low QoS where QMP and QAComPS were effective.

6.2.4. Analysis of Results

The results for AHP showed it performed well for higher level of QoS while it did not perform as well for others. However AHP was more sensitive to changes to the user requests. In contrast QMP performed well for low QoS ratings but was not very good for the higher QoS level. This was due to the way QMP prioritizes the providers offering lower cost over those offering higher QoS with higher cost. One factor might be the filter as there was no filtration for the QMP and AHP. This affected the performance of AHP which returned different sized providers for low and medium QoS.

6.3. Evaluation experiment using Cloud providers

6.3.1. Objective

The objective of this experiment was to evaluate the effectiveness of the QaComPS selection process while using actual Cloud provider’s information. The initialization and maintenance of QoS was also evaluated by the experiment.

6.3.2. Setup

The experiment was undertaken using four Cloud providers and twenty five controlled user request (see table 6.2). The profiles of the four providers are given in Appendix C.

		Amazon Ec2	**FlexiScale**	**GoGrid**	**Rackspace**
Memory (GB)	Small	1.7	2	2	2
	Medium	7.5	8	8	8
	Large	15	8	12	16
	Very Large	23	-----------	24	32
CPU (a unit equates to a core with 1-1.2 GHz)	Small	1	1	1	1
	Medium	4	4	4	8
	Large	8	8	8	16
	Very Large	33.5	-----------	24	32
Non Persistent Storage (GB)	Small	160	80	80	80
	Medium	850	320	640	320
	Large	1690	1000	1000	620
	Very Large	1690	------------	800	1000
Cost	Small	0.080	0.0805	0.057	0.1216
	Medium	0.320	0.3416	0.4000	0.4861
	Large	0.640	0.4472	0.784	0.9722
	Very Large	1.30	------------	1.10	1.66

Table 6.3: VM descriptions

Table 6.3 contains the VM descriptions for the four Cloud providers. It is evident from the table that providers offer different sized VMs. The physical cost (what user pays) is in US $ per hour.

- **Reputation**

The value for reputation was initialized through Google Trends (Choi and Varian 2009) and would be updated through user reviews. Google trends is a record of the number of visits to a website. Figure 6.9 shows the number of visit to the four providers during the past year. What is interesting to see is that Rackspace has a higher rating than Amazon. This can be explained in the context the Amazon was already an established name and users did not rely on Google to access Amazons web site which was not the case for Rackspace.

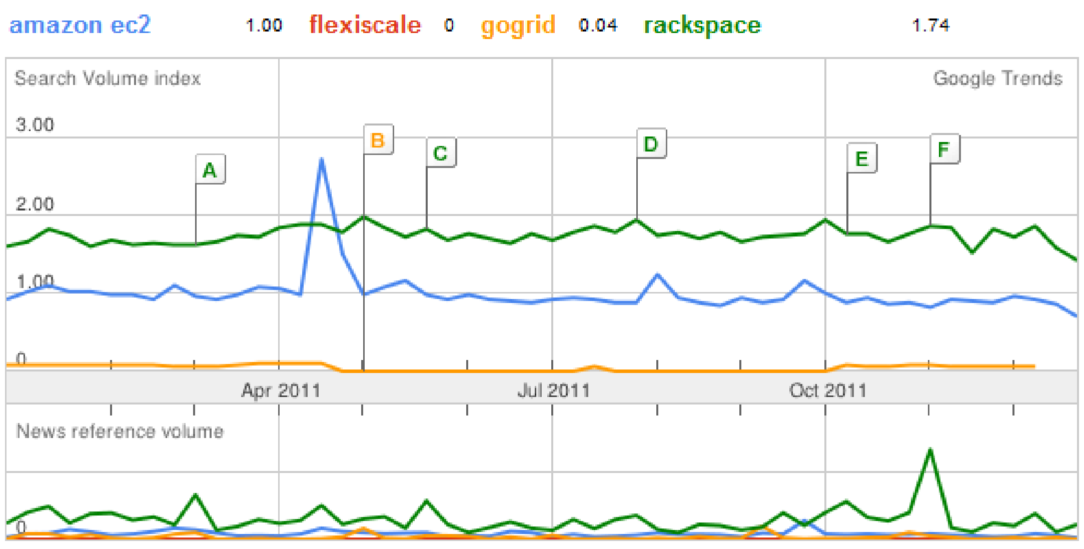


Figure 6.9 Search Trends Indicating Reputation of Cloud Providers

- **Cost**

Cost was calculated using the following equation

Cost $= (\alpha \times C_l \ + \beta \times Dt)/\ (\alpha + \beta)$

Where α and β are constant, C_l is the cost of the virtual machine and is the data transfer cost. The cost given in table 6.3 is converted into relative cost using the above equation.

Provider Name	Cost/ hour	Data transfer cost /GB	Relative Cost
AmazonEC2Small	0.080	0.12 download, free upload: relative data transfer cost = 0.08	*α= 0.65* *β=0.35* Cost=0.108
AmazonEC2Medium	0.320	0.12 download, free upload: relative data transfer cost = 0.08	*α= 0.7* *β=0.3* Cost=0.344
AmazonEC2Large	0.640	0.12 download, free upload: relative data transfer cost = 0.08	*α= 0.75* *β=0.25* Cost=0.666
AmazonEC2vLarge	1.30	0.12 download, free upload: relative data transfer cost = 0.08	*α= 0.8* *β=0.2* Cost=1.316
GoGridSmall	0.057	0.20 download, free upload: relative data transfer cost = 0.14	*α= 0.65* *β=0.35* Cost=0.106
GoGridMedium	0.4000	0.20 download, free upload: relative data transfer cost = 0.14	*α= 0.7* *β=0.3* Cost=0.442
GoGridLarge	0.784	0.20 download, free upload: relative data transfer cost = 0.14	*α= 0.75* *β=0.25* Cost=0.819
GoGridvLarge	1.10	0.20 download, free upload: relative data transfer cost = 0.14	*α= 0.8* *β=0.2* Cost=1.128
FlexiScaleSmall	0.0805	0.1 (upload & download)	*α= 0.65* *β=0.35* Cost=0.1115
FlexiScaleMedium	0.3416	0.1 (upload & download)	*α= 0.7* *β=0.3*

			Cost=0.3716
FlexiScaleLarge	0.4472	0.1 (upload & download)	*α= 0.75* *β=0.25* Cost=0.4722
RackspaceSmall	0.1216	0.14 download, free upload: relative data transfer cost = 0.09	*α= 0.65* *β=0.35* Cost=0.1531
RackspaceMedium	0.4861	0.14 download, free upload: relative data transfer cost = 0.09	*α= 0.7* *β=0.3* Cost=0.5131
RackspaceLarge	0.9722	0.14 download, free upload: relative data transfer cost = 0.09	*α= 0.75* *β=0.25* Cost=0.9947
RackspacevLarge	1.66	0.14 download, free upload: relative data transfer cost = 0.09	*α= 0.80* *β=0.20* Cost=1.68

Table 6.4: QoS Cost Table

The cost QoS is calculated by identifying the smallest value in a group; for example 0.106 for GoGrid small VM. This would be assigned highest rating of 9. The other provider's VMs would be assigned values relative to this rating. For example AmazonEC2Small that has a rating of 0.108 with a difference of 0.002 with the GoGrid small is also assigned high QoS rating. FlexiScale has a rating of 0.115 and is assigned medium rating while Rackspace being the most expensive was assigned low QoS rating. Table 6.5 contains the full list of cost QoS ratings.

- **Reliability**

  The value for reliability is based on SLA on offer from providers, literature and technical support. Commercial Cloud providers are very reliable and failures are a rare occurrence. Therefore identifying the reliability data through experimentation was not feasible as it could potentially take a year between failures. The SLA is part of the reliability data as it represents the amount of failures which will not be covered by the providers. This figure is very low as all providers offer at least 99% SLA meaning that in less 1% of the cases the user will not be reimbursed. The values are also based on the technical support; as this is vital in case of failures.

Amazon offers 99.95% SLA which is high however a lack of free technical support means that it gets a rating of 7. FlexiScale offers 100% SLA and also offers free technical support. During the past two years FlexiScale did suffer a major service failure which lasted a whole day. Therefore reliability rating for FlexiScale is 6. GoGrid offers 100% SLA and free customer support however they also suffered some service issues over the past two years. GoGrid is rated 7 for reliability. Rackspace offers a variable SLA depending on the failure. The user would be reimbursed 5% to a 100% of the credit depending on the size of the failure. This does not help the user as even a small failure could mean restarting the job from scratch. Rackspace has been given a reliability rating of 6.

- **Security**

The security rating is a measure of the security offered by providers. It is measured by analyzing the security offered by each provider. Each provider is analyzed for access and data protection. The process was carried out with the help of literature. Access is given a higher weight as it can lead to data failures.

Amazon offers a multi layer security setup which includes certificates and keys therefore it gets the highest rating of 9. GoGrid uses RBAC which is a reliable method of access control however compared to Amazon it does not use combinations of security keys; it has a rating of 5. FlexiScale does not employ RBAC or security keys but does offer each customer a personal Virtual LAN; it gets a rating of 6. Rackspace offers the same (username/password) security as FlexiScale therefore it is also rated 6.

Name	**Cost**	**Reliability**	**Reputation**	**Security**
AmazonSmall	9.0	7.0	9.0	9.0
AmazonMedium	9.0	7.0	9.0	9.0
AmazonLarge	6.0	7.0	9.0	9.0
AmazonvLarge	8.0	7.0	9.0	9.0
GoGridSmall	9.0	7.0	5.0	5.0
GoGridMedium	6.0	7.0	5.0	5.0
GoGridLarge	3.0	7.0	5.0	5.0
GoGridvLarge	9.0	7.0	5.0	5.0
FlexiSmall	7.0	6.0	3.0	6.0
FlexiMedium	8.0	6.0	3.0	6.0
FlexivLarge	9.0	6.0	3.0	6.0
RackspaceSmall	3.0	6.0	7.0	6.0
RackspaceMedium	3.0	6.0	7.0	6.0
RackspaceLarge	3.0	6.0	7.0	6.0
RackspacevLarge	6.0	6.0	7.0	6.0

Table 6.5: Initial QoS values

6.3.3. Results

Evaluation experiments were carried out using twenty five user requests as shown in table 6.2. Queries for small and medium VM returned AmazonEC2 as not only was it the cheapest provider; it had the highest QoS ratings.

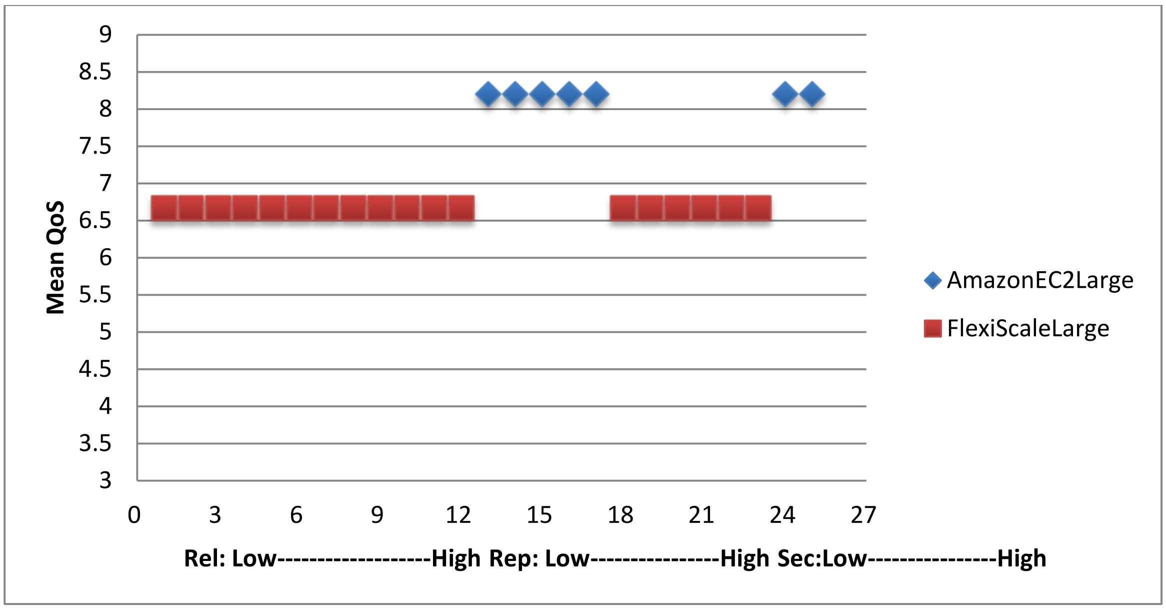
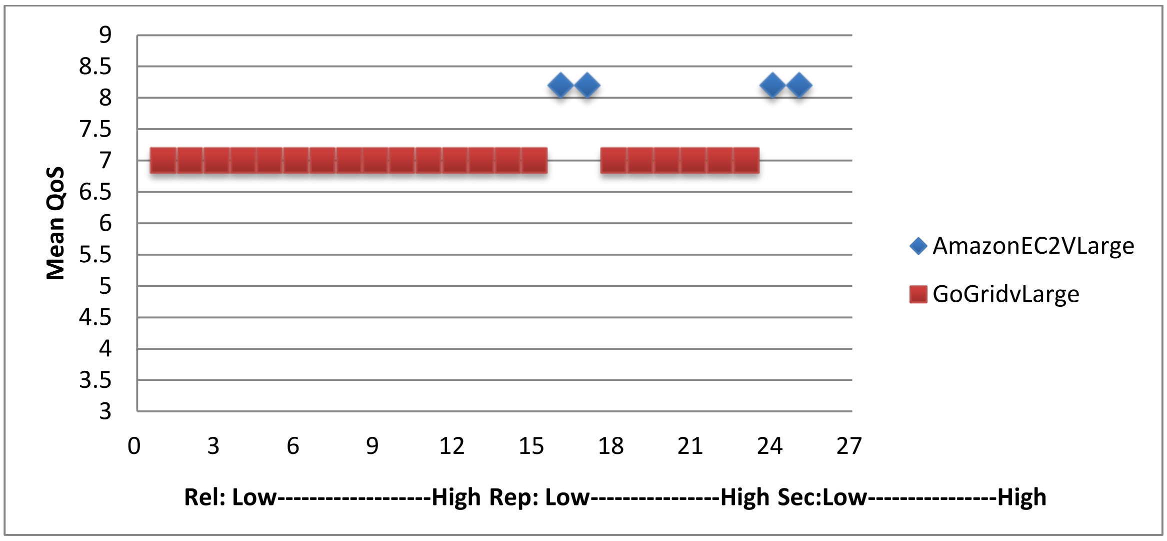


Figure 6.10: Results for Large VM



Graph 6.11: Results for very Large VM

The selection results for large VM were dominated by FlexiScale as it was the only provider in the group that had a rating of 9 for cost. FlexiScale was selected low, medium, high reliability; low reputation; and low, medium security. AmazonEC2 was selected in other cases due to a cost rating of 8.0 with the highest reliability, reputation and security ratings.

The selection of very large VMs was dominated by GoGrid due to its highest rating for cost. GoGrid was selected for low, medium reliability, reputation and security. AmazonEC2 was selected for high reputation and security.

6.3.4. Analysis of Results

As expected the results were dominated by Amazon EC2 which was the most cost effective and had a high reputation and security rating. The reputation rating would be the be most dynamic rating as after each transaction a user rates the service.

Further experimentation is needed to understand the impact on the QoS parameters and how they normalize over a longer period of time. In addition a sensitivity analysis to test the mapping of the values should also be part of the future work. The mapping translates physical values into QoS and changes to the mapping would potentially affect the results.

6.4. Early User Evaluation

An early user evaluation of the QaComPS was undertaken with the help of a computational modeller (user). The aim was to record user's perspective of the QaComPS service. This recording would help identify any deficiencies in the service. The process was be based on user scenarios, semi structured interviews, discussion, prototype demonstration and user exploration of the prototype. This was performed by one computational modeller (potential user). The aim of the evaluation was to identify how the QaComPS would be used and how could it be improved. Objectives for the evaluation are given below:

- Does a practising computational modeller feel that switching to Cloud computing would be beneficial?

- Does the user feel that QaComPS makes it easier for him to use the Cloud; by automatically selecting and running the cheapest Cloud provider on his behalf?

- Does a practising computational modeller feel that there is sufficient value in the QAComPMS to justify the overhead of providing QoS ratings and QoS weights?

6.4.1. Evaluation Process

The evaluation process employed qualitative methodology which involves using scenarios. The evaluation process was initiated by presenting the scenario, followed by a demonstration of the QaComPS service. In the end a semi structured interview was conducted to record user's views of the service. The service was evaluated subjectively by one computational modeller who are part of the HPC group at the University of Leeds.

Three user scenarios were developed (see Appendix B) for describing the usability and benefits of QaComPS service. These scenarios were later amalgamated to produce the scenario given in chapter 3. The first scenario presents the need for switching to Cloud computing, over the existing in-house setup. The second scenario presents the need for a service to identify the cheapest provider as the number of providers and options available can be time consuming and baffling. The third scenario presents the case for the use of QoS parameters and how their usage can improve the selection process.

The scenarios were followed by a demonstration of the QaComPS service. The user was given the chance to use the service; support was at-hand during this process.

In the final step a semi structured interview was conducted to record user’s views of the QaComPS. The discussion also looked back at the user’s needs and analyzed whether these could be fulfilled by the Cloud.

6.4.2. Evaluation Results

During the evaluation the user requested a small VM with the lowest cost while other QoS parameters were kept at low. QaComPS setup a small AmazonEC2 VM and passed the public DNS information along with the security key onto the user, these enabled the user to access the VM. The client machine was a laptop with an 2.5 Ghz dual core processor and 4 gb of memory. The universities high speed wireless internet connection was used for accessing the VM. The modeller was solving an Elasto Hydrodynamic Lubrication (EHL) problem for which he had previously generated a mesh on his local machine.

During the initial briefing means of accessing the VM and alternatives were described to the user. The VMs are accessible only through the static IPs and DNS keys shared by the QaComPS. The access was monitored by the QaComPS as it was responsible for the billing of the service usage.

The modeller was also briefed about the alternative solution that being bypassing the QaComPS and accessing the providers directly. The benefit of this approach being in direct contact with the VM provider however the lack of QoS data meant that the user had no knowledge regarding providers past also the user would have develop an understanding of each individual provider and share his debit/credit card details with every one of them.

Key responses from the computational modeller regarding his experience are as follows:

- According to the expert the move to the Cloud computing would help his work as it saves time and effort. “Using Cloud resources would save us time as queuing for resources was time consuming”.

- The expert stated that the selection service was easy to use and the running of the selected provider was seamless.

- The modeller was impressed with the response time which included setting up and running the VM. This time was less than two minutes. (Amount of time between making a request and getting a running VM).

- Proposed improvements were regarding running batch jobs as in the current system every job is treated individually.

- The other concern was with the storage of data as in the current setup there were no provisions for persistent storage of information. “Research experiments are performed iteratively where one set of results are inputs to the next. There should be a provision to save the results for long periods of times as these would be required in the future”.

6.5. Conclusion

The chapter has presented the evaluation results for the QaComPS; these show that QaComPS was able to select the cheapest provider consistently.

Three quantitative evaluation experiments were carried out to monitor the performance of QaComPS. All of the experiments were carried out using controlled input data set. The data set is crucial in identifying the progress of the service as results for the controlled data set are compared against results from the QaComPS and other services. A match is recorded as a “hit” while if the two sets do not match it is recorded as a “miss” and the service is debugged for errors. Two of the experiments used simulated provider’s while the third used actual Cloud providers.

The qualitative evaluation was very limited as only two experts were available for the process. It is proposed that in the future a through qualitative evaluation study should be conducted.

Chapter 7
Conclusion and Future Work

This PhD presented a study of the design and evaluation for the proposed Quality aware Computational Cloud Selection (QaComPS) service. The service addresses the High Performance Computing (HPC) needs of computational modellers by selecting and running the best Cloud provider's VM.

The thesis has proposed a new semantic approach to describing Cloud providers. The descriptions follow a service ontology for specifying the Cloud entities and their relations. At the moment each Cloud provider has its own vocabulary however ontology descriptions mean that the descriptions are uniform. The thesis also proposes four quality metrics for keeping track of a Cloud provider's progress. Considering that the Cloud providers offer a best effort solution these QoS metrics are especially useful for the computational modeller. The novel architecture for the QaComPS service has a five step selection process. This includes input, filter, rank, select, and run steps.

This chapter concludes the thesis. It is distributed into two sections where; section 7.1 describes the research findings while section 7.2 describes the potential future research work.

7.1. Research Findings

The summary of the research findings is given below:

(1) From the requirements analysis process three main groups of requirements were identified. These formed the motivation for the design of the envisioned system. The requirements include; (a) requirements related to the acquiring of HPC resources; (b) requirements related to the descriptions of Cloud providers and automatic management of provider information; and (c) requirements related to the effectiveness and efficiency of the search process.

(2) The Semantic Web is an effective technology for describing functional and Quality of Service information. The semantic data model adds meaning to the Cloud providers descriptions. This enables the QaComPS to enforce a common vocabulary while describing providers.

(3) SAWSDL annotations are effective for sharing information between semantic and non semantic sources. In case of this work the Information Management service was semantic while the selection and run services were non semantic; the SAWSDL lifting and lowering schema mappings enabled these to intercommunicate.

(4) The filter is used for filtering out providers that do not meet the user's requirements. This is particularly useful for filtering out providers that do not meet the user's functional requirements.

(5) QaComPS selection service promises higher levels of precision as the combination of ranking and selection algorithms produce effective search results. Both the algorithms can potentially be individually used to find the best result therefore their combination would reaffirm the results. The processing times were be reduced by minimizing the number of potential providers through filtering and ranking.

(6) QaComPS enables the users to access a number of Cloud providers through a single user account thus removing the chances of getting locked onto a single provider. This is achieved by using flexible mapping and programming APIs.

7.2. Future Work

The possible future work for this research includes; (a) future work for extending the QaComPS; (b) getting closer to a “guaranteed” solution; and (c) application of QaComPS as an integrated component to a large Cloud brokering system.

7.2.1. Future Work for QaComPS

The future work for the QaComPS includes extending QaComPS, improved service automation, and further evaluation.

7.2.1.1. Extending QaComPS

In the current setup the user has a fixed set of VMs to choose from however some providers such as Rackspace and GoGrid enable the user to create customized VMs with user specified processing, memory and storage. The QaComPS can be extended to include availability of the option for the customized VMs.

The concepts related to Cloud providers QoS information have never been completely identified. In this research the author selected four QoS parameters only for illustrative purposes. These can be extended as potentially there are more than 30 parameters that could be monitored (Tran, Tsuji et al. 2009). Extending QoS parameters may enhance the decision making ability of the QaComPS.

7.2.1.2. Improving Service Automation

In the current setup QaComPS automatically maintains the QoS information however a lab manager is required to keep track of the cost and the functional information. The lab manager is also responsible for finding new providers and adding their information to the database.

The process of finding new providers and updating functional information can potentially be automated by a Web crawler (Boldi, Codenotti et al. 2004). A Web crawler can search the web on-behalf of other services; it is commonly used by search engines to update information.

The incorporation of SAWSDL annotations means that the information can be processed by semantic software agents. Software agents are automatic software applications that can automatically invoke themselves for performing a set task. These can potentially, be used to update the provider information automatically.

7.2.1.3. Further Evaluation

An important next step is to fully evaluate the run service with the help of a large set of computational modellers. This includes comparing the in-house HPC machines and the Cloud providers. This evaluation provides new insights to design of the next generation of brokers wishing to offer this type of service.

The evaluation used only four actual Cloud providers. The number of potential providers can be increased to extend the evaluation process.

Another aspect for extending the evaluation would be to increase the number of VM sizes. In the current evaluation four VM sizes were used however providers like AmazonEC2 offer up to 17 different VM sizes.

7.2.2. Getting Closer to a "Guaranteed" Solution

An interesting research problem is to investigate how to get closer to a "guaranteed" solution. A guaranteed solution enables the user to calculate with absolute certainty the amount of time it would take a job to finish on a particular VM; regardless of any failures or performance lags. In order to achieve this QaComPS would require, means for continuous performance monitoring and check-pointing.

Checkpoints are traditionally used for recovery at times of service failures. In this research however, they would be used to monitor the performance of Cloud provider's VM. Given that a job takes many hours it would be possible to switch to another VM if the job run fell behind some expected schedule. This would require running more than one provider's VMs in parallel (replication). At each checkpoint the service selects the provider's VM that competes first for the next step. The chances of service failure are reduced exponentially by as the job is running on multiple VMs. A negative for this approach would be the potential higher cost as the user has to pay for all the VMs running the job.

7.2.3. Further Applications of QaComPS

The QaComPS service architecture could potentially be used for other domains. This is possible due to the independent internal service architecture as each service has a specific set of inputs and outputs regardless of the domain. In order to use another domain; new service ontology and RDF descriptions would be required while the selection service would not require many changes. The Run service is however designed specifically for working with the Cloud providers.

This research has the potential to assist brokers in providing a better service by making independent assessment of the QoS values. A Cloud Broker mediates between a provider and a user. Cloud service brokerage is an emerging field and a number of Cloud brokers have been developed (Smith 2012). However building and maintaining any broker service requires a significant investment. Today the cost can be significantly reduced by using a number of specialised third party Cloud services. One such service is the QAComPS.

Figure 7.1 shows one possible use of the QAComPS as a plug-in to a Cloud brokering system. Accesses to the QAComPS service is via the Broker service interface as shown in figure 7.1. The other service is the Data Cloud (for persistent storage of data) for managing the data on behalf of the modeller. The Application service provides the computational model that would be used by the QaComPS.

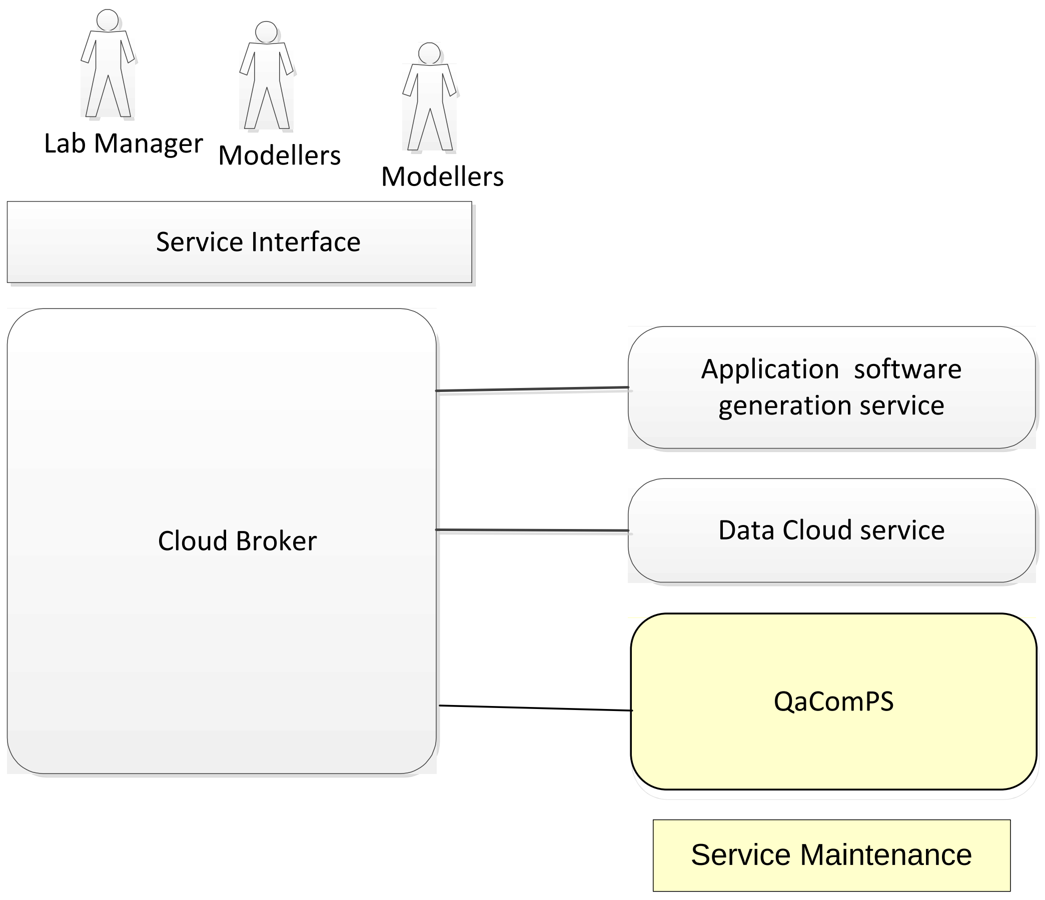


Figure 7.1: Application of the QAComPS service

The QAComPS service can also be integrated into the OPTIMIS framework which is a holistic approach to Cloud service provisioning (Ferrer, Hernandez et al. 2010). Its goal is to enable organizations to automatically externalize services to trustworthy and auditable Cloud providers in a hybrid Cloud model.

List of References

Akioka, S. and Y. Muraoka (2010). HPC Benchmarks on Amazon EC2. Advanced Information Networking and Applications Workshops (WAINA), 2010 IEEE 24th International Conference on.

Akkiraju, R., J. Farrell, et al. (2005). "Web service semantics-WSDL-S." W3C Member Submission 7.

Altintas, I., C. Berkley, et al. (2004). Kepler: an extensible system for design and execution of scientific workflows, Los Alamitos, CA, USA, IEEE Comput. Soc.

Amazon. (2012). "Amazon EC2 Service Level Agreement." Retrieved 21/03/2012, 2012, from http://aws.amazon.com/ec2-sla/.

Amazon. (2012). "Amazon Elastic Compute Cloud." Retrieved 19/03/2012, 2012, from http://aws.amazon.com/ec2/.

Anderson, D. P., J. Cobb, et al. (2002). "SETI@ home: an experiment in public-resource computing." Communications of the ACM 45(11): 56-61.

Armbrust, M., A. Fox, et al. (2010). "A view of cloud computing." Communications of the ACM 53(4): 50-58.

Armstrong, D. and K. Djemame (2009). Towards Quality of Service in the Cloud. in Proc. of the 25th UK Performance Engineering Workshop, Leeds, UK.

Arshad, J., P. Townend, et al. (2009). Quantification of security for compute intensive workloads in clouds, IEEE.

Bader, D. A. and J. JáJá (1999). "SIMPLE: A methodology for programming high performance algorithms on clusters of symmetric multiprocessors (SMPs)." Journal of Parallel and Distributed Computing 58(1): 92-108.

Bailey, D. H., E. Barszcz, et al. (1991). The nas parallel benchmarks summary and preliminary results, IEEE.

Balani, N. (2005). "The future of the Web is Semantic." IBM developerworks.

Beazley, D. M. (2009). Python essential reference, Addison-Wesley Professional.

Berners-Lee, T., J. Handler, et al. (2006). "The semantic Web." Database and Network Journal 36(3): 7-14.

Berners-Lee, T. and J. Hendler (2001). "Scientific publishing on the semantic web." Nature 410: 1023-1024.

Boag, S., D. Chamberlin, et al. (2003). "XQuery 1.0: An XML query language." W3C working draft 12.

Böck, H. and J. Tulach (2009). The Definitive Guide to NetBeans Platform, Springer.

Boldi, P., B. Codenotti, et al. (2004). "Ubicrawler: A scalable fully distributed web crawler." Software: Practice and Experience 34(8): 711-726.

Box, D., D. Ehnebuske, et al. (2000). Simple object access protocol (SOAP) 1.1, May.

Bratt, S. (2007). "Semantic web, and other technologies to watch." World Wide Web Consortium: January.

Bray, T., D. Hollander, et al. (1999). "Namespaces in XML."

Bray, T., J. Paoli, et al. (1997). "Extensible markup language (XML)." World Wide Web Journal 2(4): 27-66.

Bruijn, J., C. Bussler, et al. (2005). Web Service Modeling Ontology (WSMO), W3C Member Submission 3 June 2005.

Buyya, R., D. Abramson, et al. (2000). Nimrod/G: an architecture for a resource management and scheduling system in a global computational grid, Los Alamitos, CA, USA, IEEE Comput. Soc.

Buyya, R., C. S. Yeo, et al. (2009). "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility." Future Generation Computer Systems 25(6): 599-616.

Camara, J., J. A. Martin, et al. (2009). Itaca: An integrated toolbox for the automatic composition and adaptation of web services, IEEE.

Cassidy, P. (2008). "Toward an Open-Source Foundation Ontology Representing the Longman’s Defining Vocabulary: The COSMO Ontology OWL Version." Ontology For The Intelligence Community: 45.

Chang, V., G. Wills, et al. (2011). Towards a structured Cloud ROI: The University of Southampton cost-saving and user satisfaction case studies. Sustainable Icts and Management Systems for Green Computing, Idea Group,U.S.

Choi, H. and H. Varian (2009). "Predicting the present with Google trends." Google inc: 1-23.

Claro, D. B., P. Albers, et al. (2006). "Web services composition." Semantic Web Services, Processes and Applications: 195-225.

Consortium, O. C. (2009). Open cloud consortium.

Consortium, U. (2000). The Unicode standard, version 3.0, Addison-Wesley Professional.

Curbera, F., M. Duftler, et al. (2002). "Unraveling the Web services web: an introduction to SOAP, WSDL, and UDDI." Internet Computing, IEEE 6(2): 86-93.

Danielsson, P. E. (1980). "Euclidean distance mapping." Computer Graphics and image processing 14(3): 227-248.

De Roure, D., N. R. Jennings, et al. (2005). "The Semantic Grid: Past, Present, and Future." Proceedings of the IEEE 93(3): 669-681.

Deelman, E., G. Singh, et al. (2008). The cost of doing science on the cloud: the montage example, Ieee.

Denzin, N. K. and Y. S. Lincoln (2011). The SAGE handbook of qualitative research, Sage Publications, Inc.

Dew, P., J. Schmidt, et al. (2003). The white rose grid: practice and experience, Citeseer.

Dikaiakos, M. D., D. Katsaros, et al. (2009). "Cloud computing: Distributed Internet computing for IT and scientific research." Internet Computing, IEEE 13(5): 10-13.

Djemame, K., J. Padgett, et al. (2011). "Brokering of risk-aware service level agreements in grids." Concurrency and Computation: Practice and Experience.

Eerola, P., B. Kónya, et al. (2003). The NorduGrid production Grid infrastructure, status and plans, IEEE Computer Society.

ELEYAN, Amna, et al. (2004). Quality-of-service support in Web services architecture. Paris, FRANCE, Lavoisier.

ELEYAN, A. L., Z. (2011). "Service selection using quality matchmaking." Communications and Information Technology (ICCIT), 2011 International Conference.: 8.

Evangelinos, C. and C. N. Hill (2008). Cloud Computing for parallel Scientific HPC Applications: Feasibility of running Coupled Atmosphere-Ocean Climate Models on Amazon's EC2. CCA-08. Chicago, ACM.

Fairley, R. E. (2007). "The influence of COCOMO on software engineering education and training." Journal of Systems and Software 80(8): 1201-1208.

Fensel, D., F. M. Facca, et al. (2011). "OWL-S and Other Approaches." Semantic Web Services: 251-278.

Ferraiolo, D., D. R. Kuhn, et al. (2007). Role-based access control, Artech House.

Ferrer, J., F. Hernandez, et al. (2010). OPTIMIS: a Holistic Approach to

Cloud Service Provisioning. Future Generation Information Technology.

FlexiScale. (2012). "FlexiScale." Retrieved 21/03/2012, 2012, from http://flexiscale.com/.

FlexiScale. (2012). "FlexiScale Pricing." Retrieved 19/03/2012, 2012, from http://www.flexiscale.com/products/flexiscale/pricing/.

Foster, I., T. Freeman, et al. (2006). Virtual clusters for grid communities, Los Alamitos, CA, USA, IEEE Comput. Soc.

Foster, I. and C. Kesselman (2001). Computational grids, Berlin, Germany, Springer-Verlag.

Foster, I., C. Kesselman, et al. (2001). The anatomy of the grid: enabling scalable virtual organizations, USA, Sage Science Press.

Foster, I., A. Roy, et al. (2000). A quality of service architecture that combines resource reservation and application adaptation. Quality of Service, 2000. IWQOS. 2000 Eighth International Workshop on.

Foster, I., Z. Yong, et al. (2008). Cloud computing and grid computing 360-degree compared, Piscataway, NJ, USA, IEEE.

Geelan, J. (2009). "Twenty-one experts define cloud computing." Cloud Computing Journal 2009: 1-5.

Geelan, J. (January 22, 2009). "100 Players in the Cloud Computing Ecosystem." Retrieved 21/01/2009, from http://cloudcomputing.sys-con.com/node/770174.

Gentzsch, W. (2001). Sun grid engine: Towards creating a compute power grid, IEEE.

Godse, M., U. Bellur, et al. (2011). "A taxonomy and classification of web service QoS elements." Int. J. of Communication Networks and Distributed Systems 6(2): 118-141.

Godse, M., U. Bellur, et al. (2011). "A taxonomy and classification of web service QoS elements." International Journal of Communication Networks and Distributed Systems 6(2): 118-141.

GoGrid. (2012). "GoGrid Cloud." Retrieved 21/03/2012, 2012, from http://www.gogrid.com/21/03/2012.

Greenberg, A., J. Hamilton, et al. (2008). "The cost of a cloud: research problems in data center networks." ACM SIGCOMM Computer Communication Review 39(1): 68-73.

Gruber, T. (2008). "What is an Ontology." Encyclopedia of Database Systems 1.

Haas, R. and O. Meixner. (2005, 11/03/2011). "An Illustrated Guide to the Analytic Hierarchy Process." from http://www.boku.ac.at/mi/ahp/ahptutorial.pdf.

Hazelhurst, S. (2008). Scientific computing using virtual high-performance computing: a case study using the Amazon elastic computing cloud. Proceedings of the 2008 annual research conference of the South African Institute of Computer Scientists and Information Technologists on IT research in developing countries: riding the wave of technology. Wilderness, South Africa, ACM: 94-103.

Herman, I. (2008). "W3C semantic web activity." W3C-Semantic Web. İnternetten 2: 2009.

Horrocks, I., P. F. Patel-Schneider, et al. (2003). "From SHIQ and RDF to OWL: The making of a web ontology language." Web Semantics: Science, Services and Agents on the World Wide Web 1(1): 7-26.

Horton, I. (2011). Ivor Horton's Beginning Java, Wrox.

Iosup, A., S. Ostermann, et al. (2011). "Performance analysis of cloud computing services for many-tasks scientific computing." Parallel and Distributed Systems, IEEE Transactions on 22(6): 931-945.

Jackson, K. R., L. Ramakrishnan, et al. (2010). Performance analysis of high performance computing applications on the amazon web services cloud, IEEE.

Jaiswal, P., S. Avraham, et al. (2005). "Plant Ontology (PO): a controlled vocabulary of plant structures and growth stages." Comparative and Functional Genomics 6(7-8): 388-397.

Jena, A. (2007). "semantic web framework for Java." Retrieved 06/09/2008, 2008, from http://jena.sourceforge.net.

Jsang, A. and R. Ismail (2002). The beta reputation system.

Koivunen, M. R. (2001). "W3C semantic web activity." Semantic Web KickOff in Finland: 27-41.

Koksalan, M., M. Karwan, et al. (1984). "An improved method for solving multiple criteria problems involving discrete alternatives." IEEE Transactions on Systems, Man, and Cybernetics 14(1): 24-34.

Kondo, D., B. Javadi, et al. (2009). Cost-benefit analysis of cloud computing versus desktop grids, IEEE.

Kopecký, J., T. Vitvar, et al. (2007). "Sawsdl: Semantic annotations for wsdl and xml schema." IEEE Internet Computing: 60-67.

Kritikos, K. and D. Plexousakis (2007). OWL-Q for semantic QoS-based web service description and discovery, Citeseer.

Lara, R., D. Roman, et al. (2004). "A conceptual comparison of WSMO and OWL-S." Web Services: 254-269.

Levitt, J. (2001). "From EDI to XML and UDDI: A brief history of web services." Information Week.

Li, A., X. Yang, et al. (2010). CloudCmp: comparing public cloud providers, ACM.

Li, S. and J. Zhou (2009). The WSMO-QoS Semantic Web Service Discovery Framework, IEEE.

Linda Northrop, P. F., Richard P Gabriel, John Goodenough, Rick Linger, Tom Longstaff, Rick Kazman, Mark Klein, Douglas Schmidt, Kevin Sullivan, Kurt Wallnau (2006). Ultra-Large-Scale Systems - The Software Challenge of the Future. Technical report Software Engineering Institute Carnegie Mellon University. W. Pollak, Carnegie Mellon.

Manola, F., E. Miller, et al. (2004). "RDF primer." W3C recommendation 10: 1-107.

Martin, D., M. Burstein, et al. (2004). "OWL-S: Semantic markup for web services." W3C Member Submission 22: 2007-2004.

Martin, D., M. Paolucci, et al. (2007). "Bringing semantic annotations to web services: Owl-s from the sawsdl perspective." The Semantic Web: 340-352.

Masinter, L., T. Berners-Lee, et al. (2005). "Uniform resource identifier (URI): Generic syntax."

Mather, T., S. Kumaraswamy, et al. (2009). Cloud security and privacy: an enterprise perspective on risks and compliance, O'Reilly Media, Inc.

Maximilien, E. M. and M. P. Singh (2004). "A framework and ontology for dynamic web services selection." Internet Computing, IEEE 8(5): 84-93.

McGuinness, D. L. and F. Van Harmelen (2004). "OWL web ontology language overview." W3C recommendation 10: 2004-2003.

Menasce, D. A. (2002). "QoS issues in Web services." Internet Computing, IEEE 6(6): 72-75.

Microsoft. (2012). "Microsoft Azure." Retrieved 21/03/2012, 2012, from http://www.windowsazure.com/en-us/.

Miller, E. (2001). "W3C Semantic Web Activity." Retrieved 04/09/2008, 2008, from http://www.w3.org/2001/sw/.

Montanari, U. (1968). "A method for obtaining skeletons using a quasi-Euclidean distance." Journal of the ACM (JACM) 15(4): 600-624.

Montella, R. (2007). Development of a GT4-based resource broker service: an application to on-demand weather and marine forecasting, Berlin, Germany, Springer.

Moreau, J. J., R. Chinnici, et al. (2006). "Web services description language (WSDL) version 2.0 part 1: Core language." Candidate recommendation, W3C.

Motik, B. and I. Horrocks (2008). OWL datatypes: design and implementation, Berlin, Germany, Springer-Verlag.

MySQL, A. (2005). MySQL: the world's most popular open source database, MySQL AB.

Nandigam, J., V. N. Gudivada, et al. (2005). "Semantic web services." Journal of Computing Sciences in Colleges 21(1): 50-63.

Nelson, A. J., G. W. Dinolt, et al. (2011). A security and usability perspective of cloud file systems. System of Systems Engineering (SoSE), 2011 6th International Conference on.

Nelson, M. L., J. A. Smith, et al. (2006). Efficient, automatic web resource harvesting, ACM.

NGS. (2012). "Cloud@NGS." Retrieved 29/05/2012, 2012, from http://www.ngs.ac.uk/accessing-the-ngs-cloud-service.

Nurmi, D., R. Wolski, et al. (2009). The eucalyptus open-source cloud-computing system, IEEE.

Ostermann, S., A. Iosup, et al. (2010). "A performance analysis of EC2 cloud computing services for scientific computing." Cloud Computing: 115-131.

Pan, J. Z. and I. Horrocks (2007). "RDFS(FA): connecting RDF(S) and OWL DL." IEEE Transactions on Knowledge and Data Engineering 19(2): 192-206.

Papaioannou, T. G. and G. D. Stamoulis (2006). "Reputation-based policies that provide the right incentives in peer-to-peer environments." Computer Networks 50(4): 563-578.

Perrey, R. and M. Lycett (2003). Service-oriented architecture, IEEE.

Peter Mell, T. G. (2011). "The NIST Definition of Cloud Computing." NIST special publication 800(145): 3.

Pratt, T. W. and M. V. Zelkowitz (1984). Programming languages: design and implementation, Prentice-Hall.

Prud'Hommeaux, E. and A. Seaborne (2008). "SPARQL query language for RDF." W3C working draft 4(January).

R.V.Guha. (2000). "rdfDB: An RDF database." Retrieved 12/02/2012, 2012, from http://www.guha.com/rdfdb/.

Rackspace. (2012). "Rackspace Cloud hosting." Retrieved 21/03/2012, 2012, from http://www.rackspace.co.uk/cloud-hosting/.

Ran, S. P. (2003). A model for web services discovery with QoS. ACM SIGecom Exchanges,, ACM Press. 41: 1-10.

Rimal, B. P., E. Choi, et al. (2009). A taxonomy and survey of cloud computing systems, Ieee.

Rochwerger, B. B., D. Levy, E. Galis, A. Nagin, K. Llorente, I. M. Montero, R. Wolfsthal, Y. Elmroth, E. Caceres, J. Ben-Yehuda, M. Emmerich, W. Galan, F (2010). "The Reservoir model and architecture for open federated cloud computing." IBM Journal of Research and Development 53(4): 1-11.

Rosson, M. B. a. J. M. C. (2002). Usability Engineering: Scenario-based development of human-computer interaction. CA, USA, Academic Press.

Ruebenacker, O., I. I. Moraru, et al. (2007). Kinetic modeling using BioPAX ontology, IEEE.

Saaty, T. L. (1980). Analytic hierarchy process, Wiley Online Library.

Saaty, T. L. (2003). "Decision-making with the AHP: Why is the principal eigenvector necessary." European journal of operational research 145(1): 85-91.

Saaty, T. L. (2005). "Analytic Hierarchy Process." Encyclopedia of Biostatistics.

Saaty, T. L. (2008). "Relative measurement and its generalization in decision making why pairwise comparisons are central in mathematics for the measurement of intangible factors the analytic hierarchy/network process." Revista de la Real Academia de Ciencias Exactas, Fisicas y Naturales. Serie A. Matematicas 102(2): 251-318.

Salesforce. (2012). "Building and running applications in the Cloud." Retrieved 21/03/2012, 2012, from http://www.salesforce.com/paas/.

Serhani, M. A., R. Dssouli, et al. (2005). A QoS broker based architecture for efficient web services selection, IEEE.

Serhani, M. A., R. Dssouli, et al. (2005). A QoS broker based architecture for efficient Web services selection. Web Services, 2005. ICWS 2005. 1: 113-120.

Sheila, A. M. (2001). Semantic Web Services. S. Tran Cao and Z. Honglei. 16: 46-53.

Sirin, E., J. Hendler, et al. (2003). Semi-automatic composition of web services using semantic descriptions.

Smith, B. and P. Grenon. (2002). "Basic formal ontology." Retrieved 21/11/2008, from http://ontology.buffalo.edu/bfo.

Smith, D. M. (2012). Cloud Services Brokerages: The Dawn of the Next Intermediation Age. Cloud Services Brokerage. Gartner.com.

Somasundaram, T. S., R. A. Balachandar, et al. (2006). Semantic-based grid resource discovery and its integration with the grid service broker, Piscataway, NJ 08855-1331, United States, Institute of Electrical and Electronics Engineers Computer Society.

Srivastava, B. and J. Koehler (2003). Web service composition-current solutions and open problems.

Sultan, N. (2010). "Cloud computing for education: A new dawn?" International Journal of Information Management 30(2): 109-116.

Sycara, K., M. Paolucci, et al. (2011). "Automated discovery, interaction and composition of semantic web services." Web Semantics: Science, Services and Agents on the World Wide Web 1(1).

T.Rajendran, Dr.P.Balasubramanie, et al. (2010). "An Efficient WS-QoS Broker Based Architecture for Web Services Selection". International Journal of Computer Applications 1(9): 79.

Taher, L., R. Basha, et al. (2005). Establishing association between qos properties in service oriented architecture, IEEE.

Teknomo, K. (2006). "Similarity measurement." Available: http:\\ people. revoledu. com\ kardi\ tutorial\ Similarity\[Accessed].

Tomaž Klančnik, B. J. B., Shahzad Nizamani, Peter Dew, Karim Djemame (2009). "Inoformation Resource Broker for Cloud Computing." INFOKOMTEH: 10.

Tran, V. X., H. Tsuji, et al. (2009). "A new QoS ontology and its QoS-based ranking algorithm for Web services." Simulation Modelling Practice and Theory 17(8): 1378-1398.

Triantaphyllou, E. (2000). Multi-criteria decision making methods: a comparative study, Springer.

Vidgen, R. (2002). "Constructing a web information system development methodology." Information Systems Journal 12(3): 247-261.

Vitvar, T., J. Kopecký, et al. (2008). Wsmo-lite annotations for web services. Proceeding ESWC'08 Proceedings of the 5th European semantic web conference, ACM portal.

W3C. (2012). "Ontologies." Retrieved 20/03/2012, 2012, from http://www.w3.org/standards/semanticweb/ontology.

W3Schools. (2012). "Why Web Services." Web Services Tutorial Retrieved 13/08/2010, from http://www.w3schools.com/webservices/ws_why.asp.

Wang, L., J. Tao, et al. (2008). Scientific cloud computing: Early definition and experience, Ieee.

Wang, S., Q. Sun, et al. (2010). "Towards Web Service selection based on QoS estimation." Int. J. of Web and Grid Services 6(4): 424 - 443.

Wang, Y. and J. Vassileva (1007). A Review on Trust and Reputation for Web Service Selection. Distributed Computing Systems Workshops, 2007. ICDCSW '07.

Zeng, L., B. Benatallah, et al. (2004). "QoS-aware middleware for web services composition." Software Engineering, IEEE Transactions on 30(5): 311-327.

Zhou, C., L. T. Chia, et al. (2004). DAML-QoS ontology for web services, IEEE.

Zhou, J. and E. Niemela (2006). Toward semantic qos aware web services: Issues, related studies and experience. Proceedings of the 2006 IEEE/WIC/ACM on Web Intelligence.

Zhou, J., E. Niemela, et al. (2007). An integrated QoS-aware service development and management framework, IEEE.

Appendix A
Examples of Provider's RDF Profile

This appendix presents the RDF descriptions for the four Cloud providers used for the evaluation process.

A.1. RDF description of AmazonEC2Small

```
<Supplier rdf:ID="AmazonEC2Small">
<DownloadCost rdf:datatype="&xsd;float">0.12</DownloadCost>
<MonitorAndRecovery
rdf:datatype="&xsd;boolean">false</MonitorAndRecovery>
<Firewall rdf:datatype="&xsd;boolean">false</Firewall>
<ComputingCost rdf:datatype="&xsd;float">0.08</ComputingCost>
<StorageCost rdf:datatype="&xsd;float">0.12</StorageCost>
<API rdf:datatype="&xsd;boolean">true</API>
<UploadCost rdf:datatype="&xsd;float">0.0</UploadCost>
</Supplier>
<owl:DatatypeProperty rdf:ID="API">
<rdfs:domain rdf:resource="#Supplier"/>
<rdfs:range rdf:resource="&xsd;string"/>
<rdfs:comment rdf:datatype="&xsd;string"
>API enables a user to connect to write software programs which can connect
to the virtual machine and perform tasks.</rdfs:comment>
```

```
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="ComputingCost">
<rdfs:domain rdf:resource="#Supplier"/>
<rdfs:range rdf:resource="&xsd;float"/>
<rdfs:comment rdf:datatype="&xsd;string"
>The cost of resources computational resources</rdfs:comment>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="ControlPanel">
<rdfs:domain rdf:resource="#Supplier"/>
<rdfs:range rdf:resource="&xsd;boolean"/>
<rdfs:comment rdf:datatype="&xsd;string"
>Control panels are offered by some suppliers. They help users in setting up
the system.</rdfs:comment>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="cost">
<rdf:type rdf:resource="&owl;AnnotationProperty"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="DownloadCost">
<rdfs:domain rdf:resource="#Supplier"/>
<rdfs:range rdf:resource="&xsd;float"/>
<rdfs:comment rdf:datatype="&xsd;string"
36
>The cost of dowloads for the virtual machine. This cost is measured in
```

```
GB/month.</rdfs:comment>

</owl:DatatypeProperty>
```

A.2. RDF Description for GoGridSmall

```
<Supplier rdf:ID="GoGridSmall">

<DownloadCost rdf:datatype="&xsd;float">0.2</DownloadCost>

<LoadBalancingCost rdf:datatype="&xsd;float">0.0</LoadBalancingCost>

<MonitorAndRecovery

rdf:datatype="&xsd;boolean">false</MonitorAndRecovery>

<Firewall rdf:datatype="&xsd;boolean">false</Firewall>

<ControlPanel rdf:datatype="&xsd;boolean">false</ControlPanel>

<LoadBalancing rdf:datatype="&xsd;boolean">false</LoadBalancing>

<ComputingCost rdf:datatype="&xsd;float">0.805</ComputingCost>

<StorageCost rdf:datatype="&xsd;float">0.06</StorageCost>

<API rdf:datatype="&xsd;boolean">false</API>

<API rdf:datatype="&xsd;boolean">true</API>

<UploadCost rdf:datatype="&xsd;float">0.1</UploadCost>

</Supplier>

<owl:DatatypeProperty rdf:ID="Firewall">

<rdfs:domain rdf:resource="#Supplier"/>

<rdfs:range rdf:resource="&xsd;boolean"/>

<rdfs:comment rdf:datatype="&xsd;string"

>Some vendors offer virtual machines with firewall installed. A firewall

secures a virtual machine while it is connected to the Internet. A virtual machine is
```

```
always connected to the internet.</rdfs:comment>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="FirewallCost">
<rdfs:domain rdf:resource="#Supplier"/>
<rdfs:range rdf:resource="&xsd;float"/>
<rdfs:comment rdf:datatype="&xsd;string"
>Charges for using firewall. These charges are per month.</rdfs:comment>
</owl:DatatypeProperty>
```

A.3. RDF Description for FlexiscaleSmall

```
<Supplier rdf:ID="FlexiscaleSmall">
<DownloadCost rdf:datatype="&xsd;float">0.1</DownloadCost>
<LoadBalancingCost rdf:datatype="&xsd;float">0.02</LoadBalancingCost>
<MonitorAndRecovery
rdf:datatype="&xsd;boolean">true</MonitorAndRecovery>
<Firewall rdf:datatype="&xsd;boolean">true</Firewall>
<ControlPanel rdf:datatype="&xsd;boolean">true</ControlPanel>
<LoadBalancing rdf:datatype="&xsd;boolean">true</LoadBalancing>
37
<ComputingCost rdf:datatype="&xsd;float">0.057</ComputingCost>
<StorageCost rdf:datatype="&xsd;float">0.3</StorageCost>
<FirewallCost rdf:datatype="&xsd;float">0.01</FirewallCost>
<API rdf:datatype="&xsd;boolean">true</API>
<UploadCost rdf:datatype="&xsd;float">0.1</UploadCost>
```

```
</Supplier>

<owl:DatatypeProperty rdf:ID="LoadBalancing">

<rdfs:domain rdf:resource="#Supplier"/>

<rdfs:range rdf:resource="&xsd;boolean"/>

<rdfs:comment rdf:datatype="&xsd;string"

>Load Balancing is an optoional service offered by some vendros. Load

Balancing balances the computing load among several virtual

machines.</rdfs:comment>

</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="LoadBalancingCost">

<rdfs:domain rdf:resource="#Supplier"/>

<rdfs:range rdf:resource="&xsd;float"/>

<rdfs:comment rdf:datatype="&xsd;string"

>Charges for using load balancing. These charges are per

month.</rdfs:comment>

</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="MonitorAndRecovery">

<rdfs:domain rdf:resource="#Supplier"/>

<rdfs:range rdf:resource="&xsd;boolean"/>

<rdfs:comment rdf:datatype="&xsd;string"

>The monitoring and recovery service is offered by some

suppliers.</rdfs:comment>

</owl:DatatypeProperty>
```

```
<owl:DatatypeProperty rdf:ID="StorageCost">
<rdfs:domain rdf:resource="#Supplier"/>
<rdfs:range rdf:resource="&xsd;float"/>
<rdfs:comment rdf:datatype="&xsd;string"
>The cost of storage on the storage servers offered by suppliers. This cost is
measured in GB/month.</rdfs:comment>
</owl:DatatypeProperty>
<owl:Class rdf:ID="Supplier"/>
<owl:DatatypeProperty rdf:ID="UploadCost">
<rdfs:domain rdf:resource="#Supplier"/>
<rdfs:range rdf:resource="&xsd;float"/>
<rdfs:comment rdf:datatype="&xsd;string"
>The cost of uploads for the virtual machine. This cost is measured in
GB/month.</rdfs:comment>
</owl:DatatypeProperty>
<owl:Class rdf:ID="User"/>
</rdf:RDF>
```

A.4. RDF Description for RackspaceSmall

```
<Supplier rdf:ID="RackspaceSmall">
<DownloadCost rdf:datatype="&xsd;float">0.14</DownloadCost>
<MonitorAndRecovery
rdf:datatype="&xsd;boolean">false</MonitorAndRecovery>
<Firewall rdf:datatype="&xsd;boolean">false</Firewall>
```

```
<ComputingCost rdf:datatype="&xsd;float">0.1216</ComputingCost>

<StorageCost rdf:datatype="&xsd;float">0.12</StorageCost>

<API rdf:datatype="&xsd;boolean">true</API>

<UploadCost rdf:datatype="&xsd;float">0.0</UploadCost>

</Supplier>

<owl:DatatypeProperty rdf:ID="API">

<rdfs:domain rdf:resource="#Supplier"/>

<rdfs:range rdf:resource="&xsd;string"/>

<rdfs:comment rdf:datatype="&xsd;string"

>API enables a user to connect to write software programs which can connect

to the virtual machine and perform tasks.</rdfs:comment>

</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="ComputingCost">

<rdfs:domain rdf:resource="#Supplier"/>

<rdfs:range rdf:resource="&xsd;float"/>

<rdfs:comment rdf:datatype="&xsd;string"

>The cost of resources computational resources</rdfs:comment>

</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="ControlPanel">

<rdfs:domain rdf:resource="#Supplier"/>

<rdfs:range rdf:resource="&xsd;boolean"/>

<rdfs:comment rdf:datatype="&xsd;string"

>Control panels are offered by some suppliers. They help users in setting up
```

```
the system.</rdfs:comment>

</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="cost">

<rdf:type rdf:resource="&owl;AnnotationProperty"/>

</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="DownloadCost">

<rdfs:domain rdf:resource="#Supplier"/>

<rdfs:range rdf:resource="&xsd;float"/>

<rdfs:comment rdf:datatype="&xsd;string"

36

>The cost of dowloads for the virtual machine. This cost is measured in

GB/month.</rdfs:comment>

</owl:DatatypeProperty>
```

Appendix B: Initial Scenarios

B1. User Scenario 1: Cloud Broker for Computational Modellers

Takes place at HPC Modelling Laboratory, University X. The main actor is Ben, a second year PhD student.

The laboratory where Ben is currently working uses the university's HPC setup. The modellers are frustrated with the service in regards to the turnaround time. As every job is submitted to a queue the wait is far worse for larger jobs (over 12 hour) which are entertained only during the weekends..

In addition the computational modellers at the university X believe that the service costs do not adequately cater for the varying service needs. There are times (for example when demonstrating to potential sponsors) when high reliability is imperative. At other times a lower level of reliability (where there is an increased risk the job has to be re-run) would be acceptable.

The Computational Modelling Laboratory has decided that Cloud computing would be a better option for addressing their computing needs. Cloud computing would provide, predictable and much more flexible on-demand computing resources with a pay as you go cost structure.

Ben has a limited budget which makes it necessary to minimise the cost of using Cloud. For this, he has to go to a number of Cloud providers during the lifetime of the model development process. He finds that this is very difficult because the Cloud providers offer a variety of options making it hard for him to select the best provider. For this reason he decides that a more cost effective solution is to a use a Cloud Broker that specialises in providing computational Cloud services to computational modellers. Ben searches the Internet and is pleased to find QaComPS. He is pleased to find that QaComPS is an on-line service which automatically selects and runs the cheapest Cloud provider's VM.

Ben is researching new mathematical models for simulating the fluid flow in a chemical reactor. The model code is developed iteratively. The iterations typically involve code development, running the code, and analysing the model output against the results from a physical experiment. If the model results are not acceptable the model is changed and re-run.

Typically Ben requires modest computational recourses with an average frequency of 4-6, one hour runs a day. He is less concerned about the reliability and the reputation and is willing to compromise QoS for minimising the cost.

As Ben's model development proceeds to create new science he needs much larger computational resources with higher levels of reliability and good reputation. However if the model is wrong there is likely to be a number of small runs prior to repeating the experiment.

Just prior to a demonstration of the latest computational modelling results to the industrial sponsors; Ben needs very large computational resources with a higher level of reliability and reputation. In this case the laboratory is willing to pay for the premium rate for 2-3 one off runs.

Figure B.1. User Scenario

Details of Ben's computational requirements are set out in table B.1 together with the required QoS ratings given in table B.2

Turnaround time	**Memory (GB)**	**Storage(GB)**	**Processor(core)**	**Download(GB/sec)**	**Upload(GB/sec)**
30 minutes	2	60	1	1.0	0.5
12 hours	4	120	2	1.0	0.5
6 hours	8	200	4	1.5	1.0

Table B.1: Ben’s Requirements

Reliability	**Reputation**	**Security**	**Cost**
2	4	3	8
5	8	5	8
8	9	8	9

Table B.2: QoS and Cost ratings

The ratings are subjective and are scored on a 9 point scale with 1 being the lowest and 9 being the highest.

B.2. User Scenario 2 (Cost based selection):

Ben has finally moved to the Cloud where he is facing a laborious task of selecting the cheapest Cloud provider’s VM. This is due to the sheer number of Cloud providers and the number of VMs offered by each provider.

Other factors hampering the selection process include unavailability of a single vocabulary; so a Small VM from Amazon is literally different than a small VM from any other provider. The different vocabulary means that while Amazon class the service a Virtual Machine (VM), FlexiScale calls the same a Virtual Server. As not only the costs are different but the physical resources being offered are also different. Another issue is the pricing structure followed by the providers as some use actual cost in dollars or pounds include/exclude the taxes while the worst case might be with FlexiScale which uses a unit based system where everything is charged in terms of units. In this case a user has to buy units in advance and calculate how many units he would consume. Ben can either manually calculate costs or use QaComPS service to find the cheapest provider. If he decides to manually calculate cost he will have to do the following:

- Understand the different resource sizes offered by each provider and compare these. Translate the cost of each provider as some use a unit based structure while others use dollars or pounds
- Incorporate the upload and download costs

As of now there are more than twenty Cloud providers which make it very time consuming to perform the above steps for each of them. The other case would be to use QaComPS service which automatically performs these steps on behalf of the user.

B.3. User Scenario 3 (QoS)

Ben can select the cheapest provider but the question remains that, is the cheapest provider the best provider. As the selection process only incorporates cost and none of QoS parameters.

QoS parameters such as reliability, reputation and security identify the performance of a Cloud provider and help in decision making. Reliability is a measure of the rate of success for a Cloud provider’s service. Reputation is the rating given to a Provider from past users which indicates the level of user satisfaction. Security is the measure of security offered from a provider as a security breach can be critical. The question remains though to what extent QoS ratings improve the selection process. This can be decided by using QaComPS and comparing the results of simulations performed earlier with the results from using actual Cloud providers.

Appendix C: Cloud Provider Profiles

C.1. Amazon EC2

Instances		**Memory**	**Compute Units**	**Storage**
Standard	Small	1.7	1	160
	Medium	3.75	2	410
	Large	7.5	4	850
	Extra Large	15	8	1690
Micro	Micro	0.613	2 (short bursts)	-------
High Memory	Extra Large	17.1	6.5	420
	Double Extra Large	34.2	13	850
	Quadruple Extra Large	68.4	26	1690
High CPU	Medium	1.7	5	650
	Large			
Compute Cluster	Quadruple Extra Large	7	20	1690
	Eight Extra Large			
Cluster GPU	Quadruple Extra Large	22	33.5	1690
High I/O	Quadruple Extra Large	60.5	35	2048

Table C.1. Amazon EC2 VMs (Amazon 2012)

Amazon Elastic Compute Cloud (EC2) is an Infrastructure as a Service (IaaS) Cloud. It offers 13 types of Virtual Machines which as shown in table C.1.

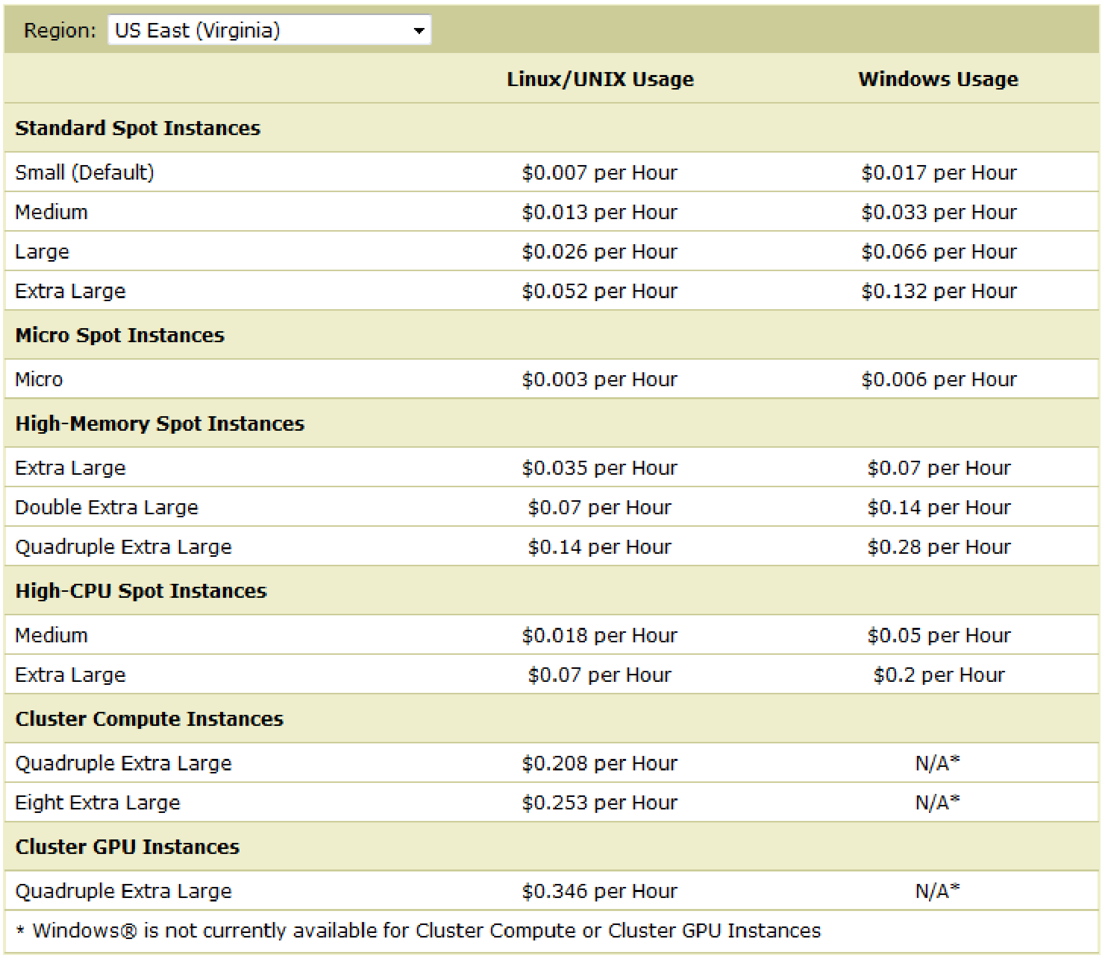
The standard instances are general purpose and suited to most applications. The micro instances are free of cost however these do not have a permanent CPU. A micro instance gets up to two EC2 compute units for short bursts of time. These contain 0.613 GB of memory and have no storage option. These are suited to low budget applications that are not on a strict deadline.

High memory instances are for running jobs that require vast amounts of memory while High CPU instances are for compute intensive jobs. The cluster compute instances are high CPU instances with extra network bandwidth. Cluster GPU instances for graphics intense applications while the I/O instances are the most powerful and suited high performance database workloads.

EC2 services are offered from seven physical zones five of which are in America, One in Ireland, One in Singapore and One in Tokyo and one in Sao Paulo. It should be noted that not all kinds of VM are on offer at each zone as only zones in America offer all the options. A VM instance is charged per-hour and the charging policy does not depend on whether it is being used or not until it is not terminated it would be charged.

AmazonEC2 has a collection of more then 7, 000 Amazon Machine Images (AMI). These are used for quickly deploying a VM. Out of these few are created by Amazon while others are created by users these are custom images and are publically shared a user can store these images without cost.

In conclusion; Amazon has many pros these include, aggressive costing, great number of options and a trusted brand while there only downside is the lack of customer support as in a normal customer is not eligible for one on one support until he signs up and pays for premium support. Other disadvantage is the added security this makes be good for a customer but this does mean that it takes longer than usual for a customer to access the newly created VM.



Region: US East (Virginia)

	Linux/UNIX Usage	Windows Usage
Standard Spot Instances		
Small (Default)	$0.007 per Hour	$0.017 per Hour
Medium	$0.013 per Hour	$0.033 per Hour
Large	$0.026 per Hour	$0.066 per Hour
Extra Large	$0.052 per Hour	$0.132 per Hour
Micro Spot Instances		
Micro	$0.003 per Hour	$0.006 per Hour
High-Memory Spot Instances		
Extra Large	$0.035 per Hour	$0.07 per Hour
Double Extra Large	$0.07 per Hour	$0.14 per Hour
Quadruple Extra Large	$0.14 per Hour	$0.28 per Hour
High-CPU Spot Instances		
Medium	$0.018 per Hour	$0.05 per Hour
Extra Large	$0.07 per Hour	$0.2 per Hour
Cluster Compute Instances		
Quadruple Extra Large	$0.208 per Hour	N/A*
Eight Extra Large	$0.253 per Hour	N/A*
Cluster GPU Instances		
Quadruple Extra Large	$0.346 per Hour	N/A*

* Windows® is not currently available for Cluster Compute or Cluster GPU Instances

Figure C.1. AmazonEC2 costing (Amazon 2012)

C.2. FlexiScale

FlexiScale is a UK based Cloud provider which offers FlexiScale Cloud service. FlexiScale offers Virtual Servers where a virtual server is the name given to a Virtual Machine. A virtual server can have between 0.5 and 8 GB of memory, up to 8 processing units and any amount of storage. The novel bit about FlexiScale is its pricing as it employs a unit based costing where a user can buy any number of units between a thousand and two million. While memory, storage, processing and data transfer is charged in terms of units for example for each GB of storage a user is charge 5 units/month this makes it expensive to use the service on a per hour basis while using the service on per month will be cheaper.

C.3. Rackspace

Rackspace is generally considered the second biggest Cloud provider after Amazon it offers services to among others Virgin Trains, London Transport, Renault, VUE, Vodafone, Dominos and NHS. They offer a competitive pricing with a large number of Cloud Servers (same as VM) and images. The unique bit about Rackspace is that in many ways it discourages per hour Cloud customers and is only focused at the monthly customers. They offer between 256 MB and 32 GB memory with any amount of storage while the processing power is unclear which might be directly proportional to the memory. Compared to Amazon, Rackspace is an expensive provider.

C.4. GoGrid

GoGrid Cloud provider offers one of the best customer-support and also offers incentives as free credit but other than that everything is going the other way. GoGrid offers very large VM with up to 32GB of memory.

Appendix D: Interview Script for Qualitative Evaluation of QaComPS

The Interview Script

The semi structured interview was the central component of the qualitative evaluation. The goal of the evaluation was to investigate:

1. User's view regarding the move towards Cloud computing.
2. Record the user's view regarding the benefits of QaComPS
3. Whether the interface of QaComPS is simple enough or does the user feel that input parameters are missing or there are inputs overheads.

Introduction

Brief the user regarding the scope of the research. This includes:

- Purpose of the research
- Purpose of this evaluation

Interview

The interview takes place in three stages; these are:

- Acquiring user's views on the current setup
- Acquiring user's view regarding Cloud computing
- Demonstrate the QaComPS prototype and record user view regarding the QaComPS

Part 1: User's view regarding the current HPC setup:

Q1: Do you have experience of using local HPC setup?

If yes; can you describe your experience?

Q2: In your view what are the short comings of the current HPC setup.

Part 2: Acquiring user's view regarding Cloud computing

Q3: What are your views on the move towards Cloud computing?

Q4: What advantages and disadvantages, do you see in using Cloud computing.

Part 3: Demonstrate the QaComPS prototype and record user view regarding the QaComPS

Q5: What benefits do you see in using QaComPS?

Q6: Do you feel that any functionality is missing from the QaComPS?

Q7: What are your views on the QaComPS interface and the user requirements?

De-brief

Thank You.

The results from the interview will be sent to you for permission before any academic publication.

Could you suggest someone else who would like to participate in the interview.