INVESTIGATION OF THE ROLE OF SERVICE LEVEL AGREEMENTS IN WEB SERVICE QUALITY

AIJAZ AHMED SOOMRO

INVESTIGATION OF THE ROLE OF SERVICE LEVEL AGREEMENTS IN WEB SERVICE QUALITY

Aijaz Soomro

A Thesis presented for the degree of

Doctor of Philosophy

School of Engineering and Computing Sciences

Durham University

January 2016

Acknowledgements

It would not have been possible for me to write this dissertation without the help and support of many people. I would like to thank to my supervisor, Professor Malcolm Munro, who generously helped me technically. Without his invaluable advice and guidance, it was almost impossible to complete this goal. Thanks Malcolm!

I am sincerely thankful to Dr. William Song for his initial guidance as my supervisor at the start of my PhD study.

How can I forget some golden words from my friend, who said ‘PhD is not a 100 meter race but it is marathon, it requires patience’. And those words really helped me to achieve this goal.

With due respect, I would like to thank my father, Naziruddin Soomro and mother Sahib Khatoon. Their support encouraged me to hit the nail of PhD target.

Lastly, but certainly not the least, I am thankful to my lovely wife, Sadaf, who always understood my tough position. Thank you for your love, support, patience, and everything!

Declaration

The work in this thesis is based on research carried out in the Innovative Computing Group, the School of Engineering and Computing Sciences, University of Durham, UK. No part of this thesis has been submitted elsewhere for any other degree or qualification. All the work presented here is the sole work of the author unless referenced to the contrary in the text.

This research has been documented or is related, in part, within the publications listed below:

- William Wei Song, Aijaz Soomro, Yang Li, Qin Liu: A Structural Analysis of SLAs and Dependencies Using Conceptual Modelling Approach. COMPSAC Workshops 2013: 439-444

- Soomro, Aijaz, and William Song. "Developing and Managing SLAs for Business Applications in Information Systems." In *Emerging Trends and Applications in Information Communication Technologies*, pp. 489-500. Springer Berlin Heidelberg, 2012.

- Soomro, Aijaz, William Song, and Yang Li. "An Investigation of the Role of Service Level Agreements in Classified Advertisement Websites." In *Computer Software and Applications Conference, 2009. COMPSAC'09. 33rd Annual IEEE International*, vol. 2, pp. 380-385. IEEE, 2009.

Dedicated to

My Father, Naziruddin Soomro

and

My Mother, Sahib Khatoon Soomro

Abstract

Context/Background:

Use of Service Level Agreements (SLAs) is crucial to provide the value added services to consumers to achieve their requirements successfully. SLAs also ensure the expected Quality of Service to consumers.

Aim:

This study investigates how efficient structural representation and management of SLAs can help to ensure the Quality of Service (QoS) in Web services during Web service composition.

Method:

Existing specifications and structures for SLAs for Web services do not fully formalize and provide support for different automatic and dynamic behavioral aspects needed for QoS calculation. This study addresses the issues on how to formalize and document the structures of SLAs for better service utilization and improved QoS results. The Service Oriented Architecture (SOA) is extended in this study with addition of an SLAAgent, which helps to automate the QoS calculation using Fuzzy Inference Systems, service discovery, service selection, SLA monitoring and management during service composition with the help of structured SLA documents.

Results:

The proposed framework improves the ways of how to structure, manage and monitor SLAs during Web service composition to achieve the better Quality of Service effectively and efficiently.

Conclusions:

To deal with different types of computational requirements the automation of SLAs is a challenge during Web service composition. This study shows the significance of the SLAs for better QoS during composition of services in SOA.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AA	-	Any Attribute
AG	-	Action Guarantee
AGR	-	Aggregation of Rules
AHP	-	Analytic Hierarchy Process
AIN	-	Agreement Initiator
ANP	-	Analytic Network Process
ANT	-	Antecedent
ARES	-	Agreement Responder
B2B	-	Business-to-Business
B2C	-	Business-to-Consumer
BPEL4WS	-	Business Process Execution Language for Web Services
C	-	Controller
C2C	-	Consumer-to-Consumer
CONS	-	Consequent
CS	-	Composite Service
DEA	-	Data Envelopment Analysis
DEMATEL	-	Decision-Making Trial and Evaluation Laboratory
ELECTRE	-	Elimination and Choice Expressing Reality
EXC	-	Exclusions
FAM	-	Fuzzy Associative Memory
FIS	-	Fuzzy Inference System
FO	-	Fuzzy Output
FOP	-	Fuzzy Operator
FUN	-	Function
GRA	-	Grey Relational Analysis
GT	-	Guarantee Terms
I	-	Inputs to Service
IaaS	-	Infrastructure as a Service
ILV	-	Input Linguistic Variable
IMF	-	Input Membership Function
IMOP	-	Implication Operator

IR	-	Input Range
IT	-	Information Technology
MADM	-	Multi-Attribute Decision Making
MCDA	-	Multiple-Criteria Decision Analysis
MCDM	-	Multi-Criteria Decision Making
MD	-	Measurement Directive
MET	-	Metric
MF	-	Membership Function
MODM	-	Multi-Objective Decision Making
NIST	-	National Institute of Standards and Technology
O	-	Outputs of Service
OASIS	-	Organization for the Advancement of Structured Information Standards
OB	-	Obligations
OLV	-	Output Linguistic Variable
OMF	-	Output Membership Function
OR	-	Output Range
ORS	-	Online Reputation System
OS	-	Optional Service
OWL-S	-	Web Ontology Language for Services
PaaS	-	Platform as Service
PAR	-	Parties
PEN	-	Penalties
PR	-	Party Roles
PROMETHEE-		Preference Ranking Organization Method of Enrichment Evaluations
QoS	-	Quality of Service
QoS_SCORE	-	Quality of Service Score
QoSD	-	QoS Database
QoSD:1	-	QoS Database:1. Store
QoSD:2	-	QoS Database:2. Retrieve
QoSM	-	QoS Monitor
QoSM:1	-	QoS Monitor:1. Service Level Measurement
QoSM:2	-	QoS Monitor:2. Monitor SLA Violation
QoSM:3	-	QoS Monitor:3. Feedback Receiver

QoSM:4	-	QoS Monitor:4. Reputation Builder
REQ	-	Requirement
RES	-	Restrictions
RP	-	Requirement Processor
RP:1	-	Requirement Processor:1. Discover Service Provider
RP:2	-	Requirement Processor:2. Discover Service Provider QoS
RP:3	-	Requirement Processor:3. Discover Composite Services with QoS
RP:4	-	Request Service Level Agreement
RP:5	-	Requirement Preferences
RS	-	Rule Strength
S	-	Service
SaaS	-	Software as a Service
SC	-	Service Consumer
SCOP	-	Scope
SDT	-	Service Description Terms
SIP	-	Signatory Party
SLA	-	Service Level Agreement
SLAED	-	SLA End Date
SLAM	-	SLA Manager
SLAM:1	-	SLA Manager:1. Define SLA Template
SLAM:2	-	SLA Manager:2. SLA Negotiation
SLAM:3	-	SLA Manager:3. SLA Establishment
SLAM:4	-	SLA Manager:4. SLA Deployment
SLAM:5	-	SLA Manager:5. SLA Termination
SLAM:6	-	SLA Manager:6. Enforce Penalties for SLA Violation
SLAPAR	-	SLA Parameters
SLAPU	-	SLA Purpose
SLASD	-	SLA Start Date
SLAVP	-	SLA Validity Period
SLO	-	Service Level Objective
SM	-	Service Manager
SM:1	-	Service Manager:1. Service Development/Preparation
SM:2	-	Service Manager:2. Service Composition

SM:3	-	Service Manager:3. Composition Filter
SM:4	-	Service Manager:4. Execution
SM:5	-	Service Manager:5. Decommission
SOA	-	Service Oriented Architecture
SOAP	-	Simple Object Access Protocol
SOP	-	Service Operations
SP	-	Service Provider
SPRO	-	Service Properties
ST	-	Service Terms
SUP	-	Supporting Party
TOPSIS	-	Technique for Order of Preferences by Similarity to Ideal Solution
TP	-	Third Party
TRS	-	Trust and Reputation System
UDDI	-	Universal Description Discovery and Integration
W	-	Weight
WS-CDL	-	Web services Choreography Description Language
WSDL	-	Web Services Description Language
WSMF	-	Web Service Modeling Framework
WSML	-	Web Service Modeling Language
WSMO	-	Web Service Modeling Ontology
WSMX	-	Web Service Execution Environment

1. Introduction

1.1 Introduction

The research contribution of this thesis is to investigate the role of Service Level Agreements in QoS during Web service composition in the context of computational services to provide adequate Quality of Service in Service Oriented Architecture (SOA). It does so by defining a model SLAAgent and using Fuzzy Logic as a measurement tool. The original contribution of the research in this thesis is an improved way of selecting services based on the Quality of Service using Fuzzy Inference System with multiple inputs. This then leads to an increased clarity of Service Level Agreements. It also contains a novel application of Fuzzy Logic to Service Level Agreements and Quality of Service.

Service providers offer their services with a claims of the minimum to maximum expected Quality of Service (QoS) they can provide. The level of QoS is normally defined in terms of a Service Level Agreement (SLA). The consumers can receive the services either with the required level of QoS or less than the expectations. The variations in the QoS can reflect the reputation of the service providers. The clarity of the SLA structures and management removes much of the uncertainty in communication between consumers and service providers. The SLAs offer service providers the ability to differentiate their services in a competitive market.

This chapter discusses the motivation of the study with a motivational scenario, problem statement, research question, thesis contribution, structure of thesis and finally the summary of the chapter.

1.2 Motivation

A Personal Computer is divided into three layers, the lowest layer is hardware known as infrastructure, the middle layer is Operating System known as platform layer and the top layer is user level known as software. While providing computing as a service over the network requires specific Web based softwares and platforms.

A service is defined as the means of delivering value to consumers by providing the outcomes that consumers want to achieve without having the ownership of specific costs and risk [85]. A service provides the set of facilities, which can be related to Information Technology (IT) or non-IT, sustained by the IT service provider that fulfils one or more requirements of the consumer, supports the consumer's objectives and is perceived by the consumer as a coherent whole [31]. A service can be composed of other services, which itself can be composed of one or more other IT based systems within a complete infrastructure [14].

In SOA, Web services are the most popular choice for implementing the services, because they are self describing and platform independent as compared to other implementations such as CORBA (Common Object Request Broker Architecture) and DCOM (Distributed Component Object Model) [123]. Web services are the distributed building blocks which present well defined interfaces that process and deliver messages for communications among them. Web service is a technology that provides a systematic and extensible framework for application-to-application interaction which is built on the top of the existing Web protocols, such as the Extensible Markup Language (XML) [43]. Due to the use of XML all hindrances caused by programming language dependencies and system dependencies are removed [110].

Figure 1-1 illustrates the SOA-based service consumer and service provider interaction.

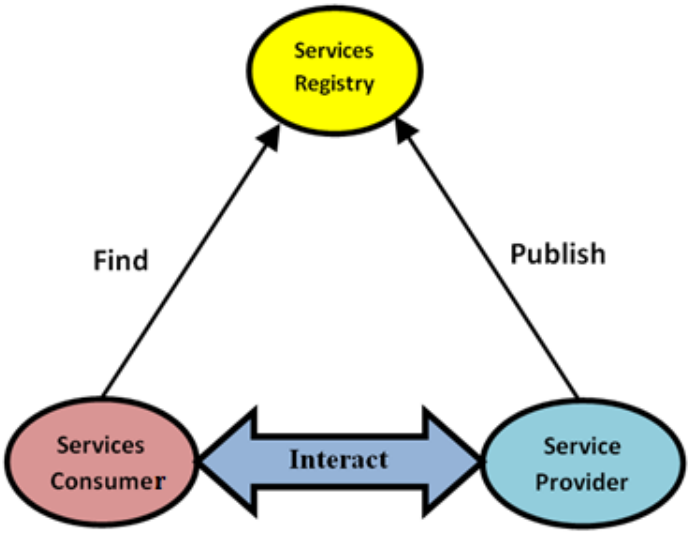

Figure 1-1: SOA

The motivation of this research is to investigate and incorporate the SLAs and QoS into SOA and in particular using computational services.

1.2.1 Motivational Scenario

A newly created company wants to utilize IT services and want to setup their IT services from different computational service providers according to their usage on demand. The company needs to use different computational services from different service providers such as Web Hosting and File Storage services. The company finds a number of functionally similar services that are available from various computational service providers with different Quality of Service rankings. Each computational service provider can have different terms and conditions of the service mostly defined in Service level Agreements (SLAs).

It is very difficult for the newly created company as a consumer to select the most useful and reliable services according to their requirements. The consumer not only needs to select the services one by one but also needs to take care of the dependency between the services such as scheduling and integrating the services according to the terms and conditions of the services that should match their company's requirements.

Due to the wide range of available computational service providers available on the internet, the selection and composition of the required services and interpreting their SLAs manually becomes very complex task. Therefore there is need to use some automated mechanism such as the Service Oriented Architecture (SOA) paradigm, in order to fulfill the requirements efficiently and effectively.

Therefore the selection of SOA based service provision of computational services using Web services and their composition methodologies specially using SLAs can be the best choice for the IT company to fulfill their requirements.

1.3 Problem Statement

Web services are the most accepted invocation of the SOA [32], but the structural description of the Web service standards are mainly developed and are particularly

suitable for service providers to describe and publish their services, and they do not support the management and monitoring aspects for SLAs and QoS. Service consumers are not given good support for automatically searching and selecting the services. Therefore, some manual effort is needed by the service consumers to select the required services from a group of functionally similar services with different Quality of Service ranking score in order to fulfill their requirements.

e-Businesses have the need to integrate their business processes and software applications with Web services over the Internet. To offer a best quality service over the Internet is a challenge because of its dynamic nature. To determine the Quality of a service it is essential for the services to have an unambiguous and formal service contract between service provider and the service consumer.

In order to achieve a satisfactory level of the services being offered, the commitments and assurances of services are implemented and assured through the use of a newly developed mechanism called the Service Level Agreements (SLAs), which establish an agreed contract between the service consumers and the service providers by stating the expectations and obligations explicitly defined between them [93].

The current main SLA approaches such as WSLA [73], WS-Agreement [18] and SLAng [79] are focused on the functional aspects of services and do not cover the detailed QoS monitoring and management aspects. Based on the literature survey of this thesis none of them has become standard. There is more need to focus on the improvement of structural and non-functional terms of the services that are important to determine the accurate overall QoS and also to build the good business relationship with the consumers by offering the value added services with distinguished features among the different service providers.

Quality of Service is a very important factor for differentiating and selecting the service of any type, but most of the work in industry and academia on QoS is focused on the computational aspects of the services, while the non-functional aspects such as general as well as domain specific characteristics of Quality of Service still need to be standardized.

Service composition is also important for utilizing various services in a group of services (composite services) for fulfilling the requirements of consumers using more than one service. The Business Process Execution Language for Web Services (BPEL4WS) and Web services Choreography Description Language (WS-CDL) are the popular Web service composition approaches for static composition, and Web Ontology Language for Services (OWL-S) and Web Service Modeling Ontology (WSMO) for dynamic Web service composition which are focused on functional composition of services. The context of Quality of Service using Service Level Agreements at the time of Web service composition is lacking in these approaches.

This study has a major focus on structuring, managing and monitoring of SLAs and non-functional characteristics for Quality of Service and its calculation in Web service composition using SLAs.

1.4 Research Question

The main research question addressed in this thesis is:

How to structure and manage Service Level Agreements automatically and effectively for value added Quality of Service during Web service composition

The sub-problems related to the main research question are:

1. *How to properly document the structure of SLAs*
2. *How to manage the SLAs*
3. *How to monitor the SLAs*
4. *How to calculate the Quality of Service*
5. *How to compose the services to fulfill consumer requirements based on QoS using SLAs.*

The research problem involves investigation of dynamic creation, management and monitoring of SLAs during the composition of required services that should be appropriate to fulfill the consumer requirements. It also involves investigating the criteria for quantifying the Quality of Services and its integration into existing Web service standards at the time of service composition. Therefore, in order to achieve the

consumer requirements, the suitable services should be selected and then composed accordingly and properly. The composition and execution of services using SLAs and QoS require the understanding of: the consumer requirements; the detailed description about the services; the Quality of Service attributes the services provide; the Quality of Service attributes the consumer requires; the detailed structure and management of Service Level Agreements and how to calculate the QoS.

1.5 Thesis Contribution

Guided by the research questions and sub-problems stated in Section 1.4, the following contributions to the state of the art in SLAs and QoS calculation during service composition research have been made. Figure 1-2 shows the illustration of the thesis contribution.

Main Contribution:

Definition of a Model for Structured SLA Management and Monitoring for Web Service Quality

Sub Contributions are:

1. *Basic Concept Elements for SLA structures*
2. *Framework Component for SLA Management*
3. *Framework Component for QoS Monitoring using SLAs*
4. *Quality of Service Calculations*
5. *Framework Component for Managing Service Composition Using QoS Information from SLAs*
6. *Advance in Fuzzy Inference Systems with more than two inputs.*
7. *Increased Clarity in Service Level Agreements*
8. *Better Implementation of Fuzzy Inference System*
9. *Improved way of selecting services based on QoS.*
10. *Application of Fuzzy Inference System to SLAs and QoS*
11. *Usage of QoS Terms within and without SLA Parameters*

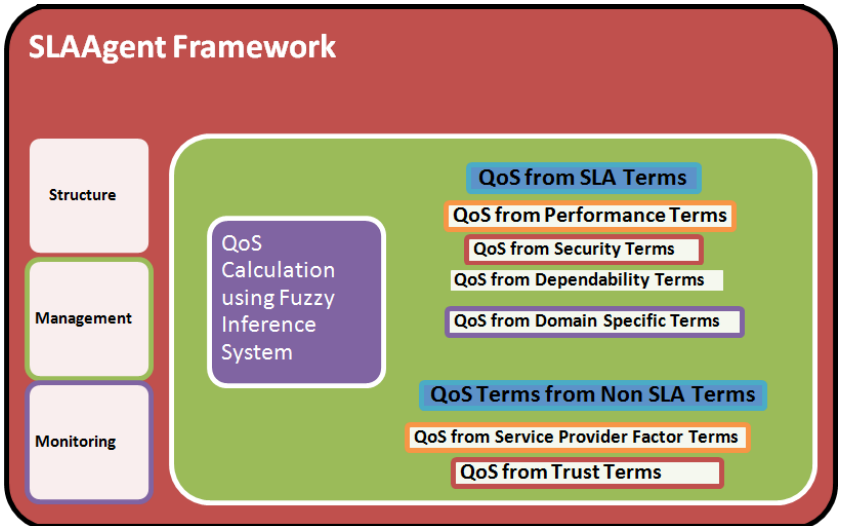

Figure 1-2 : Thesis Contribution Illustration

1.6 Structure of the Thesis

Chapter 2 is the state of the art review, which discusses the various approaches related to thesis problem. Chapter 3 is based on writing the definition for basic concept elements used for formalizing, managing and monitoring the SLAs and related Quality of Service terms. It also includes the Ontological representation of SLA Elements and QoS Terms used in the thesis. Chapter 4 derives the formulas for calculating the Quality of Service based on the use of a Fuzzy Logic Inference System. In Chapter 5 the proposed framework for structured SLA management and monitoring for QoS is discussed in detail. Chapter 6 provides use case scenario from computational services that utilizes the proposed framework from chapter 5. Chapter 7 evaluates the proposed framework against the use case scenario and comparative evaluation with other related approaches. Chapter 8 concludes the thesis and discusses about the future directions of the research that can be carried out based on this thesis.

1.7 Chapter Summary

This chapter introduced the concepts of SLAs and QoS within SOA environment. A motivational scenario was discussed which created the inclination towards the solution of research problem. The research question and its related sub problems have been discussed in this chapter that will be addressed throughout the thesis. The thesis contribution has been discussed in detail and illustrated in pictorial representation followed by the structure of thesis.

2. State of the Art

2.1 Introduction

In the beginning of the World Wide Web (WWW), its main purpose was to provide the data and information to people. However, later on the needs changed, and people or businesses required the same information in machine-readable form. Web services are a solution to such requirements and allow software applications to access the required information by connecting applications to applications to achieve specific tasks without the problem of platform heterogeneities. Web services are the most popular choice for implementing the services in a Service Oriented Architecture (SOA). All data and meta data is transferred using Extensible Markup Language (XML), hence the programming language and systems dependency is removed, with the help of Web services [32].

People and businesses are moving towards the use of computing as a service which is also known as Cloud computing. Service consumers may require more than one service from the same or different service providers at the same time. Manually it is very difficult for consumers to discover, select and buy the multiple services separately, particularly those services which are tightly inter-related to each other. There are different Web service composition approaches from research and academia who have tried to solve the problems of automatic Web service composition. However, selecting and composing Web services based on Quality of Service (QoS) using Service Level Agreements (SLAs) is a big challenge in the field of Web services. The use of properly defined SLA structures, their management and Quality of Service calculation is at the heart of this thesis.

This chapter reviews the standards used in Service Oriented Architecture (SOA), Web service components, the background and related approaches for Service Level Agreements, SLA lifecycle, Trust and Reputation Systems, computational services, Quality of Service and Web service composition. It also reviews the state of the art to characterizes the SLA material, aggregates the highly relevant SLA elements, restructures the SLA elements and then refines the SLA structure in a well-classified form. This chapter surveys the cloud computing and its most popular service providers

along with their monitoring tools support. The various QoS metric terms from different approaches are gathered and refined in this chapter. This chapter also surveys the QoS section techniques, advantages of Fuzzy Inference System and its use by different approaches.

2.2 Service Oriented Architecture

The term “Architecture” is formally defined as a system, which includes its purpose, functions, interfaces and externally visible properties. It also contains the detail about the internal system components and their relationships along with specific rules followed by its design, operations and evaluation. A service is defined as a software unit or component that can be accessed and used over a network to provide the functionality to the requester of the service [111].

A Service-Oriented Architecture (SOA) provides a paradigm for building distributed applications. In SOA, services are distributed elements, which provide well-defined interfaces that process and deliver messages for communications. The scope of a service based approach helps with building cross-organizational applications. A business with multiple systems and applications on different platforms can benefit from SOA to build a loosely coupled integrated solution that implements unified workflows [63].

2.2.1 Web Services

According to W3C [25], “A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format. Other systems interact with the Web service in a manner prescribed by its description using Simple Object Access Protocol (SOAP) messages, typically conveyed using Hyper Text Transfer Protocol (HTTP) with an XML serialization in conjunction with other Web-related standards.”

Web services are software entities which are self-contained and loosely coupled components. Web services can be published, located and invoked across the web. They offer a technique for building interoperable, distributed and platform as well as language independent systems. The Web services are designed to be incorporated in a

SOA paradigm inherently. Their features assure immediately the requirements that services in a Service-Oriented Architecture should satisfy [78].

A Web service is described by a Web Service Description Language (WSDL) document. The service provider publishes the Web service into the Universal Description Discovery and Integration UDDI repository. A client application searches the UDDI registry and discovers the required service. The client obtains the WSDL document, and a Simple Object Access Protocol (SOAP) request message is generated based on the WSDL document. The Web service request handler parses the SOAP message, then invokes the right Web service, and creates the SOAP response. It finally sends the response back to the client.

2.2.2 Web Service Components

2.2.2.1 Web Service Description Language

Web Service Description Language (WSDL) [36] is an XML based interface description language for Web services. The service providers specify the operations of a Web service into a WSDL document. The WSDL also describes the parameters and data types of the operations supported by services. It contains all the mandatory information that helps consumers to interact with services, such as message formats, transport protocols and location of services. WSDL hides the details of the implementation from service consumers, so that the services can be used as hardware and software independent. The platform independence allows the Web services to work as loosely coupled distributed, SOA based software solutions.

2.2.2.2 Universal Description, Discovery and Integration

Universal Description, Discovery and Integration (UDDI) [39] describes a set of data structures to represent Web services for the intention of advertisement and discovery. Therefore, the service providers can publish the important information about the businesses, services offered and protocols for the communication to UDDI so that the service consumers can look into the repository of services and select the required services to buy a single service or they may compose more than one service for their requirement.

2.2.2.3 Simple Object Access Protocol

A WSDL file contains all the information needed to describe and invoke the services through the Simple Object Access Protocol (SOAP) [58]. SOAP is a platform neutral network transport protocol that allows consumers to call a remote service. SOAP is designed to provide communication between systems on different platforms. The building blocks of a SOAP document contains 1) the SOAP envelope, which defines the namespace and the encoding style, 2) the SOAP header, which defines other characteristics of the message, and 3) the SOAP body, which contains the data of the message that is being sent.

2.3 Service Level Agreements

A Service Level Agreement (SLA) is a legal requirement for any business to have some formal contracts for the services they offer to their customers. An SLA is an agreed upon statement of the expectations and obligations that exist in a business relationship between the user and the service provider. It is a formalized representation of commitments in the form of an SLA document which is required when the information is collected and SLA evaluations are to be automated [94].

An SLA is a contract which is related to the guarantees of a Web service. An SLA defines the level of service formally. SLAs describe the common understandings about services, priorities, responsibilities, guarantees, level of service availability, performance, and operation. An SLA is a formally negotiated agreement between two parties. An SLA between a service provider and its customers will assure customers that they can get the service for which they are paying and it will obligate the service provider to obey its service promises [68]. The service guarantees are about what transactions need to be executed and how well they should be executed.

2.3.1 Service Level Agreement Elements

There are different SLA structures defined by various industries and academia, but to the best of our knowledge none of the approach has become the standard for detailed QoS monitoring and management aspects, therefore they still need improvements. This

thesis has summarized the set of SLA elements from the highly relevant approaches [73], [18] and [68] which are shown in Table 2-1, Table 2-2 and Table 2-3 respectively.

There are also some approaches that have organized the management of agreements (SLAs) into two categories i.e. single-layer and multiple layers [59]. The approaches that manage agreements at single-layer are discussed in (e.g. [82], [61], [127], [23] and [76]), and the approaches that deal the management of agreements in multiple layers i.e. business layer, service layer and infrastructure layer are discussed in (e.g. [40], [117], [118] and [27]). All these single-layer and multiple layer approaches listed are inspired from, and are sub sets of major approaches [18], [73] and [68].

From Table 2-1, Table 2-2 and Table 2-3 it is clear that there is no agreed standard and harmony among them. The elements from these tables are collected together and shown in Table 2-4. From Table 2-4 only two elements (i.e. 30 and 31) are present in all three of the approaches surveyed, five elements (i.e. 25, 26, 27, 28 and 29) are present in two of the approaches in two different combinations out of three of them. While the elements (1-12, 13-20 and 21-24) are uniquely present in these three approaches. Table 2-5 shows the SLA elements rearranged and restructured by this thesis that were taken from collection of elements shown in Table 2-4. After restructuring the SLA elements from different approaches, the SLA elements are further refined by this thesis, which are shown in Table 2-6.

Table 2-1 : Service Level Agreement Elements from WS-Agreement [18]

S.No	SLA Elements
1	Agreement Name
2	Agreement Context
3	Agreement Terms
4	Service Terms
5	Guarantee Terms
6	Agreement Initiator
7	Agreement Responder
8	Service Provider

9	Service Consumer
10	Agreement Expiration Time
11	Agreement Template
12	Any Attribute
13	Service Description Terms
14	Service References
15	Service Properties
16	Service Scope
17	Qualifying Condition
18	Service Level Objective
19	Business Value List

Table 2-2 : Service Level Agreement Elements from WSLA [73]

S.No	SLA Elements
1	Service Definition
2	Service Object
3	SLA Parameter
4	Metric
5	Measurement Directive
6	Function
7	Parties
8	Signatory Party
9	Supporting Party/Third Party
10	Obligations
11	Guarantee
12	Service Level Objective
13	Action Guarantee

Table 2-3 : Service Level Agreement Elements from HP by Jin [68]

S.No	SLA Elements
1	Purpose
2	Parties
3	Validity Period
4	Scope
5	Restrictions
6	Service Level Objectives
7	Penalties
8	Optional Services
9	Exclusions
10	Administration

Table 2-4 : SLA Elements Collected Together

S.No	SLA Elements	Reference
1	Agreement Context	[18]
2	Agreement Terms	
3	Agreement Initiator	
4	Agreement Responder	
5	Service Provider	
6	Service Consumer	
7	Agreement Template	
8	Any Attribute	
9	Service References	
10	Service Properties	
11	Qualifying Condition	
12	Business Value List	
13	Service Object	[73]
14	Signatory Party	
15	Supporting Party/Third Party	
16	SLA Parameter	
17	Metric	
18	Measurement Directive	

19	Function	
20	Obligations	
21	Restrictions	[68]
22	Optional Services	
23	Exclusions	
24	Administration	
25	Service Definition/ Service Description Terms/Service	[73][18]
26	Parties	[73][68]
27	Purpose/Agreement Name	[18][68]
28	Agreement Expiration Time/Validity Period	
29	Scope/Service Scope	
30	Guarantee/Action Guarantee/Guarantee Terms/Penalties	[73][18][68]
31	Service Level Objectives	

Table 2-5: Rearranged and Restructured SLA Elements in This Thesis

Rearrange/ Restructure	**S.No/SLA Elements/ (WS-Agreement)** [18]	**S.No/ SLA Elements/ (WSLA)** [73]	**S.No/ SLA Elements/ (H.P Labs)** [68]
Name	1: Agreement Name 2: Agreement Template 3: Agreement Terms		
Purpose			1: Purpose
Context	4: Agreement Context		
Validity Period	5: Agreement Expiration Time		2: Validity Period
Parties	6: Service Provider 7: Service Consumer	1: Parties	3: Parties
Party Role	8: Agreement Initiator 9: Agreement Responder	2: Signatory Party 3: Supporting Party/Third Party	
Service Terms	10: Service Terms 11: Service Description Terms 12: Service References 13: Service Properties 14: Any Attribute	4: Service Definition 5: Service Object 6: SLA Parameter 7: Metric 8: Measurement Directive 9: Function	
Guarantee Terms	15: Guarantee Terms 16: Service Scope 17: Qualifying Condition 18: Service Level Objective 19: Business Value List	10: Obligations 11: Guarantee 12:Service Level Objective 13: Action Guarantee	4: Scope 5: Service Level Objectives 6: Optional Services 7: Restrictions 8: Exclusions 9: Penalties 10: Administration

Table 2-6: Refined SLAs Elements in this Thesis

Refined SLA Elements Sections	**SLA Elements**
Name	Agreement Name Agreement Template Agreement Terms
Purpose	Purpose
Validity Period	Validity Period
Parties	Service Provider Service Consumer Third Party
Party Role	Signatory Party Supporting Party Agreement Initiator Agreement Responder
Service Terms	Service Description Terms Service Properties SLA Parameter Metric Measurement Directive Function Any Attribute
Guarantee Terms	Obligations Service Scope Service Level Objective Action Guarantee Penalties Optional Services Restrictions Exclusions

2.3.1.1 Explanation of SLA Elements:

The rearranged and restructured SLA elements are divided into sections such as Name, Purpose, Validity Period, Parties, Party Roles, Service Terms and Guarantee Terms. These elements are explained in Table 2-7, Table 2-8, Table 2-9, Table 2-10, Table

2-11, Table 2-12 and Table 2-13 respectively. The list of SLA elements not included in this thesis after rearrangement and restructure are also explained in Table 2-14.

Table 2-7: List of SLA Elements: Name

Agreement Name: The Agreement Name is optional which can be assigned to an Agreement. The Agreement Name is not dependent on the name of the template used for the Agreement. It is also not a unique identifier. It may be easily understandable by humans, it could be additional information to the Endpoint Reference of an Agreement Resource used in a protocol [18].
Agreement Template: The Agreement Template is a document defined in XML or any other Language used for Agreement, to explain the details about the offer of the service provider. It may contain the Name of the Agreement, Context of the Agreement, Agreement Terms and all the relevant information used to create the agreed actions between service provider and service consumer[18]. It can also contain particular ID or code of the SLA.
Agreement Terms: The Agreement Terms are based on one or more definitions of service Terms, and guarantee terms arranged using logical grouping operators in the quantity of zero or more[18].

Table 2-8: List of SLA Elements: Purpose

Purpose: The purpose describes the actual motive of creating the SLA [68].

Table 2-9: List of SLA Elements: Validity Period

Validity Period/Agreement Expiration Time: When an agreement is finished, this time is called as Agreement expiration time or validity period of Agreement and after that the parties involved into the agreement are not obligated any more time by the terms of the agreement [18][68].

Table 2-10: List of SLA Elements: Parties

Service Provider: The role of service provider is to provide the service to the consumer according to the conditions explained in the agreement [18].
Service Consumer: The role of the service consumer is to obtain the service and receive the guarantees of the service being provided from the service provider [18].
Third Party: The Third Parties are the supporting parties of an SLA which can be sponsored by one or both Signatory Parties (service provider/service consumer) [73].

Table 2-11: List of SLA Elements: Party Roles

Supporting Parties: The supporting party role is representation of a party role of an SLA which are not performed by main parties of the SLA. The supporting party role is also known as Third Party role of an SLA [73].
Signatory Parties: The signatory party roles of an SLA represent the main parties and they are required to sign the SLA and remain responsible for all the liabilities [73].
Agreement Initiator: The role of the Agreement Initiator is to create and manage an agreement for the service offered on behalf of either the service provider or service consumer. This role can be either performed by service provider or service consumer depends on domain specific requirement [18].
Agreement Responder: The role of the Agreement Responder is to implement and expose an agreement for the service offered on behalf of either the service consumer or service provider. This role can be either performed by service consumer or service provider depends on domain specific requirement [18].

Table 2-12: List of SLA Elements: Service Terms

Service Definition/Service Terms/Service Description Terms: Service Definition describes the operations, service parameters and metrics that are the basis of the Service Level Agreement. It may also describe the specification of the measurement of service's metrics. The service definition may include the reference to the service operations and bindings of the service defined in the guarantees of an Agreement for

the service being offered [73]. The information required for instantiation or identifying a service is provided in Service Terms. Service Terms explain the relevance of the Agreement used and the guarantee terms being applied. Service terms are further classified as Service Description Terms, Service References and Service Properties. Service Description Terms (SDTs) are the information about the functionality being delivered and explicitly explained in the Service Agreement document. SDTs are fundamental elements of an Agreement. The Service Descriptions Terms are dependent on the specific domain. SDT contains the name of the SDTs, Name of service being offered and domain specific description of the service for the offered or required functionality [18].
SLA Parameters: SLA parameters are the properties of a Service Object. Each SLA parameter has a name, type and unit. Each SLA parameter refers to one composite Metric [73].
Metric: Metrics are details about the values of Service Properties that are measured from service providing system or can be computed from other metrics and constants. Metrics are an important tool used to describe exactly what SLA Parameters mean and they specify how to measure or compute the parameter values [73].
Measurement Directive: The Measurement Directives explain how parameter values should be measured that are provided by organizations in the form of metrics for the use in SLAs. The values of measurement are totally dependent on the type of the system in which measurement is required [73].
Function: A measurement algorithm or formula is defined in a Function that explains how a composite metric is computed. The mean, median, sum, and other arithmetic operations are the examples of formulas used in functions [73].
Service Properties: Service Properties are optional. Service properties explain the domain specific features of a service, which are used to express the non-functional requirements (guarantees) of the service. Service properties are used to define measureable and exposed properties associated with a service, such as response time and throughput. These properties are used to express Service Level Objectives [18].
Any Attribute: Any Additional attribute can be specified in an Agreement but it should not contradict the actual meaning of the agreement defined by the owner of the Agreement [18].

Table 2-13: List of SLA Elements: Guarantee Terms

Service Level Objectives: Service Level Objective (SLO) describes an objective that must be fulfilled in order to provide a service with a particular Quality of Service (QoS) also called Level of Service. The level of Quality of Service is described in Service Description Terms of an Agreement. SLOs determine a logical expression that can be monitored in order to determine the fulfillment of a guarantee[18].The Service Level Objective defines the level of the service that both the consumer and service providers agree on [68]. Service Level Objectives describe the assurances with respect to the state of SLA parameters. A Service Level Objective defines a promise to maintain a particular state of the service in a particular time. SLOs give a formal expression of the guaranteed condition of a service in a given period. SLOs are the main obligations of service providers not of Supporting Parties [73].
Guarantee/Action Guarantee/Guarantee Terms/Penalties: Guarantee is defined as predicates over SLA Parameters. The value of these parameters can be obtained from the measurement function. The condition evaluation function must implement the relevant predicates to perform the guarantee evaluation. In the case of a guarantee violation, an action is invoked on the management function [73]. An action guarantee describes a commitment to carry out a particular activity if a given precondition is fulfilled. Action Guarantee mostly relates to the supporting parties of the contract and the service Customer. Action guarantees are the promises of a signatory party to perform an action. It can be any SLO violation or any trigger from management operation [73]. The service Levels that the parties are agreeing to are specified by the guarantee terms. The Guarantee Terms are used by Management Systems to monitor the service and endorse the agreement [18]. If the service provider is unable to provide the service for the expected level of service which does not meet the objectives of the SLA, then some penalties will occur. The penalties can be any but already defined by service provider in the SLA of service [68].
Obligations: The obligations define various guarantees and constraints that may be imposed on SLA parameters. The obligations define the Service Level that is guaranteed with respect to the SLA Parameters defined in the Service Definition section[73].
Scope/Service Scope: The Scope of services determines which services in the

Agreement are covered under the guarantee. Since different set of services can be part of a single Agreement, therefore each guarantee may apply to one or more services [18][68].
Restrictions: Restrictions define the necessary prohibition of some steps or actions that must not need to be taken to ensure that the requested level of service is maintained [68].
Exclusions: Exclusions explicitly specify what is not covered in the Service Level Agreement[68].
Optional Services: The services that are not mandatory and they are not part of the SLA, but requested on demand. The Optional Services can be considered as an exception [68].

Table 2-14: List of SLA Elements Not included in this Thesis

Agreement Context: The Agreement Context is a mandatory element in the Agreement, which gives the information about the Agreement that is not explained in the terms of the agreement, such as which parties are involved in the Agreement, what service has been agreed in the Agreement, the total period of the Agreement[18].
Service Object: Service Object is an abstraction of a service usually defined by the Web Service Description Language (WSDL) and Simple Object Access Protocol (SOAP), whose properties are related to defining guarantees of SLAs in the form of SLA parameters. A Service Object can contain one or more SLA Parameters and its corresponding Metrics [73].
Service References: Service Reference is optional. A Service Reference contains a domain-specific reference to an existing service. A Service Reference points to a service by providing an Endpoint Reference [18].
Administration: The administration follows the procedures defined in the SLA to meet and measure its objectives, and implementing the organizational responsibility for taking care of each of those procedures [68].
Business Value List: The Business Value List specifies the penalties and rewards that are listed in guarantee terms. The predefined business values of a guarantee are represented in value of particular currency [18].
Qualifying Condition: The Qualifying Condition determines the preconditions that must be fulfilled before any guarantee to be enforced. It is not necessary that all guarantees may apply

during the whole lifetime of an Agreement. Only the Qualifying condition can identify the preconditions that must be fulfilled before a guarantee is evaluated [18].

2.3.2 Service Level Agreement Lifecycle

The SLA structure, management and monitoring process requires various interaction steps. The formal SLA lifecycle stages reviewed from highly relevant approaches [73], [128] and [13] are shown in Table 2-15, Table 2-16 and Table 2-17 respectively. All their stages are collected together in Table 2-18 in order to show the harmony and differences between them. The approach [73] has five stages, the approach [128] has six stages and approach [13] has five stages. In Table 2-18 the collection shows stage (S.No: 11 and 12) are present in all three approaches, stages (S.No: 7 to 10) are present in two approaches but with different combination of two out of total three approaches. While stage (S.No:1), stages (S.No: 2-3) and stages (S.No: 4-6) are present individually in three different approaches [128], [73] and [13] respectively.

Table 2-19 shows the SLA lifecycle stages rearranged and restructured by this thesis that were taken from collection of SLA lifecycle stages from Table 2-18. After rearranging and restructuring the SLA lifecycle stages from different approaches, the SLA lifecycle stages are further refined by this thesis which are shown in Table 2-20.

Table 2-15 : SLA Lifecycle Stages from WSLA [73]

S.No	SLA lifecycle Stages
1	SLA Negotiation and Establishment
2	SLA Deployment
3	Service Level Measurement and Reporting
4	Corrective Management Actions
5	SLA Termination

Table 2-16 : SLA lifecycle Stages from Sun Microsystems Data Centre [128]

S.No	SLA lifecycle Stages
1	Discover Service Providers
2	Define SLA
3	Establish agreement
4	Monitor SLA violation
5	Terminate SLA
6	Enforce penalties for SLA violation

Table 2-17 : SLA lifecycle Stages TM Forum, SLA Handbook Solution Suite [13]

S.No	SLA lifecycle Stages
1	Service and SLA Template Development
2	Negotiation
3	Preparation
4	Execution
5	Assessment
6	Termination and Decommission

Table 2-18 : The collected SLA Lifecycle Stages from different Approaches

S.No	SLA lifecycle Stages	Reference
1	Discover Service Provider	[128]
2	SLA Deployment	[73]
3	Service Level Measurement	
4	Service Development/ Preparation	[13]
5	Execution	
6	Decommission	
7	Define SLA / SLA Template Development	[128] [13]
8	SLA Establishment/ Establish Agreement	[73][128]
9	Corrective Management Actions/ Enforce Penalties for SLA Violation	

10	SLA Negotiation/Negotiation	[73][13]
11	Monitor SLA Violation/ Reporting/ Assessment	[73][128][13]
12	SLA Termination/ Terminate SLA/ Termination	

Table 2-19: Restructured SLA Lifecycle Stages

Restructured SLA Lifecycle Stages	**S.No: SLA lifecycle Stages/ Reference** [73]	**S.No: SLA lifecycle Stages/ Reference** [128]	**S.No: SLA lifecycle Stages/ Reference** [13]
Discover Service Provider and Develop Service		1: Discover Service Providers	1: Service Development
Define SLA Template		2: Define SLA	1: SLA Template Development
SLA Negotiation	1: SLA Negotiation		2: Negotiation
SLA Establishment	1: SLA Establishment	3: Establish agreement	3: Preparation
SLA Deployment	2: SLA Deployment		
Execution			4: Execution
Monitor SLA Violation	3: Service Level Measurement and Reporting	4: Monitor SLA violation	5: Assessment
Enforce Penalties for SLA Violation	4: Corrective Management Actions	5: Enforce penalties for SLA violation	
SLA Terminate	5: SLA Termination	6: Terminate SLA	6: Termination
Decommission			6: Decommission

Table 2-20: Refined SLA Lifecycle Stages in this Thesis

Refined SLA Lifecycle Stages
Define SLA Template
Service Development/ Preparation
SLA Negotiation
SLA Establishment
SLA Deployment
Execution
Service Level Measurement
Monitor SLA Violation
Enforce Penalties for SLA Violation
SLA Termination
Decommission

2.3.2.1 Service Level Agreement Lifecycle Stages Explanation

The SLA lifecycle stages collected in Table 2-18 are explained in Table 2-21.

Table 2-21: SLA Lifecycle Stages Explained

Discover Service Provider: For a good SLA it is necessary to discover the resources i.e. services that could satisfy the requirements of the service consumer [128].
Define SLA/ SLA Template Development: Once service providers have been found, it is mandatory to identify the various elements of an SLA that model the required Quality of Service then the Agreement should be signed by Signatory Parties to join an Agreement [128][13].
SLA Negotiation/Negotiation: Negotiation provides a mechanism when a consumer and provider exchange a number of contract messages in order to reach a mutual Agreement. The result of these dialogues leads to a new SLA. Negotiation is the exchange of offers and counter offers between the consumer and provider. Negotiation is helpful in order to ensure that there is no conflict between the service provider and consumer in reaching an Agreement [128][13].
SLA Establishment/Establish Agreement: In this process the SLA template is created, parties negotiate if required and finally accomplished by signing the SLA by both Signatory Parties (provider and consumer). The process of an SLA being

negotiated and the signatures done by both Signatory Parties is called SLA establishment [73][128].

Service Development/Preparation: This phase includes the identification of the services that are required to consumers. The appropriate characteristics and parameters of service required in the service are also identified in this stage [13]. The service is prepared in this phase so that the specific or special consumer requirements (if any) should be covered. Therefore, some reconfigurations of the resources of service may occur in this phase in order to meet the SLA parameters [13].

SLA Deployment: The SLA deployment is liable to check the validity of the Service Level Agreement and its distribution either in whole or in some parts to the involving components. In deployment process first, the SLA deployment system of signatory party creates and sends information of configuration to its supporting parties. Secondly the deployment system of supporting parties decides to configure implementations in their own suitable way [73].

Execution: This is the process for delivering operational services to the service consumer. In this stage the actual operation of service starts [13].

Service Level Measurement: The Service Level Measurement maintains information about the current system configuration, also the runtime information about the metrics which are part of the SLA. It measures the SLA parameters. It may measure all or a subset of the SLA parameters [73].

Monitor SLA Violation/ Reporting/Assessment: Monitoring the obligations defined in the Service Level Agreement to ensure that all the agreement clauses have been fulfilled or violated by either one of the parties or both of them. SLA violation monitoring starts once the agreement has been established. It identifies which party is responsible of violation and how the satisfaction can be assured between the agreement parties [128]. It is responsible for comparing measured SLA parameters against the thresholds defined in the SLA and notifying the management system. This can be done each time a new value is available or periodically [73]. It includes the assessment of SLA and QoS that is provided to consumers. QoS, consumer Satisfaction, potential improvements, and changing requirements are reviewed periodically in this stage. The assessment of the overall service tends to review the internal business clause. The elements offered to be reviewed with respect to QoS

provided to consumers. This may also include the realignment of the service requirements and operations if necessary [13].
SLA Termination/Terminate SLA/Termination: An SLA should mention the condition under which the SLA may be terminated or certain penalties may incur on a party by breaking one or more SLA clause. There may be the chance of negotiation being carried out between the parties similar to the way at the time of SLA establishment. An SLA may also be terminated if the expiry date of the SLA is due [73][128][13].
Decommission: The decommissioning is a controlled process used to safely stop all the functionalities of a service or services that are no longer needed [13].
Corrective Management Actions/ Enforce Penalties for SLA Violation: If the service provider is unable to provide the service for the expected level which does not meet the objectives of the SLA, then some penalties will occur. Once the condition evaluation or reporting has determined that a Service Level Objective (SLO) has been violated, certain corrective management actions need to be carried out. The Management Service upon receipt of a notification will retrieve the appropriate actions to correct the problem, as specified in the SLA. But before acting upon the managed system, it consults the Business Entity to verify if the proposed actions are allowable [73][128].

2.3.3 Service Level Agreement Approaches

2.3.3.1 Web Service Level Agreement (WSLA)

The Web Service Level Agreement for Web Services (WSLA) [73] is a project from IBM for Specifying and Monitoring Service Level Agreements for Web services. WSLA addresses issues and challenges of service level management in Web services environment related with SLA specification, its creation and monitoring. It can calculate and monitor Quality of Service characteristics and inform of any violations to the participating parties. It is based on formal XML schema language to represent SLAs and the framework to interpret the Language of framework at run time. It supports separate monitoring section in addition to the agreement terms for outsourcing purposes. It is based on a language which can specify SLAs that can be monitored by the customer, service provider and by a third-party. It supports the creation of SLA

templates, and it has separate monitoring framework for distributed environment. In this framework new metrics can be created based on the existing metrics to support the various QoS Parameters, but the context of the metrics are not formally supported in this framework, therefore it becomes very difficult for creating the new terms based on existing terms.

2.3.3.2 Web Service Level Agreement Specification (WS-Agreement)

The Web Service Agreement Specification (WS-Agreement) [18] is a proposal of the Open Grid Forum [11] defined by GRAAP Working Group [9]. It is the standard for specification and creation of SLAs known as the Web Service Agreement Specification. The main role of WS-Agreement Specification is to provide a protocol and language for marketing the capabilities of service providers and generates the agreements between consumers and providers support for negotiating, managing and monitoring the agreements at runtime. WS-Agreement is based on Extensible Markup Language (XML) supported language for creating the agreement template, and supports the feature for discovery of compatible providers. It works in request and response mode. WS-Agreement allows the parties to expose their status, so that any SLA violation could be managed and verified dynamically. Initially the support for negotiation was not available in the language but later on it has been added into it. Negotiation is based on the top layer of WS-Agreement and supports the re/negotiation of the SLA.

2.3.3.3 A Language of Defining Service Level Agreements (SLAng)

A Service Level Agreement Language (SLAng) [79] uses XML for defining the SLAs. Its main objective is to provide the specification for creating the contractual relationship between the Customers and Application service providers for stating the clear statement about the obligations on all participating parties involved into the SLA with respect to predefined QoS. SLAng initially defines the vocabulary of SLA for Internet Services. Later on it creates the structure based on industry specific requirement for providing the usable terms. Finally it is designed to use the Unified Markup Language (UML). Its definition is created according to the behaviour of services and consumers participating in the use of service. It also supports the scheme for third party monitoring. It does not have the ability to define the management information for example associated financial terms. Hence it may not be appropriate for commercial type computing environments.

2.3.3.4 Web Service Offering Language (WSOL)

Web Service Offering Language (WSOL) [120] is a compatible notation with WSDL (Web Service Description Language) using XML. WSDL describes the operations provided by Web services, while WSOL allows the formal specification of multiple classes of service for a Web service. Mixing WSOL descriptions with standard WSDL descriptions, a Web service can be described in more detail regarding QoS, cost of service and other non-functional constraints. WSOL supports template instantiation and reuse of definitions. WSOL allows formal specification related with functional constraints, Quality of Service constraints, access rights, cost and the interaction with other service offerings from the same Web service provider. Specifying a Web service with the help of WSOL, along with WSDL, helps in the selection of more suitable Web services and offerings of service for particular requirements. WSOL also supports the adaptation and management of Web service compositions dynamically with the help of service offerings manipulation. WSOL provides the way of specifying replacement of service offerings, if the agreed offerings cannot be fulfilled.

2.3.3.5 Rule-Based Service Level Agreement

The Rule-Based Service Level Agreement (RBSLA) [95] is the specification language for SLAs used in electronic services. The RBSLA is an extension of XML-based Rule Markup language RuleML [24]. The RuleML is mainly based on rules describing the Resource Description Framework (RDF) [84]. RDF is a language used to describe resources, their properties, relationships and their types with the help of XML and XML Schemas. The RBSLA has a core contribution for contracts and Service Level Management tools as a rule based language which describes the contracts or policies formally in Service Level Agreements. The knowledge representation concepts of the approach are drawn with the help of Artificial Intelligence (AI) and also with the help of Web service standards and Semantic Web technology. The core uses of the RBSLA are to investigate the expressive logic programming techniques and logical formalisms. These techniques include defeasible, deontic, temporal action, truncation, update and description logics as a mean of deriving formal and declarative contract specifications. These logics could also help to reason about the actual behaviors of the contracts such as permissions, obligations, prohibitions, violations and exceptions to the contracts. The

RBSLM (Rule-Based Service Level Management) tool is the implementation of RBSLA which is being built on ContractLog KR [96] and an open source rule engine Prova [77]. The RBSLA approach provides the benefits of automated verification, validation and consistency check of possibly large distributed and interchangeable rule sets, an automated chaining, reasoning and execution of rules, distributed contract modules and flexible dynamic extension of new contract rules.

2.3.3.6 Generalized Service Level Agreement (GSLA)

A Generalized Service Level Agreement (GSLA)[15] is a an agreement signed between two or more parties belonging to a service relationship who intend to build unambiguous, measureable and general understanding for each party role involved in the Service Level Agreement. The set of rules defined by a party role includes service level obligations and service level expectations along with constraints. The constraints are represented with particular types such as the scope of contract, the billing policies which are agreed mutually and in case of abnormal service operation the predefined remedies. A language specification of Service Level Agreements also called as (GXLA) [116] is the implementation of GSLA model. GXLA maintains the multi-party relationship of services using a role-based mechanism. All kinds of IT business relationships are covered in GXLA for complex service interactivity using SLA modeling. GXLA is based on XML schemas which provide the general ground base between the entities for the automation of the configuration. In the configuration of IT systems, GXLA can be used by service consumers, service providers and Third Parties. GXLA is role-oriented, while each role includes a set of Service Level Objectives and rules which characterize each party's behavior in the SLA. The SLAs in GXLA are composed of *Schedule*, *GXLAParty*, *ServicePackage* and *Role*.

2.3.3.7 Quality of Service Modeling Language (QML)

The Quality of Service Modeling Language (QML) [50] is a language for defining multi-category QoS specifications for components in distributed object systems. The general QoS support is provided by QML with main focus on reliability, performance, security and timing. QML is used to describe QoS properties in software components. It is a kind of interface definition language which not only describes the functional characteristics of software components, but also has the capability to define the non-

functional properties of software components. QML supports static decomposition of software components into precisely specified QoS boundaries. It also facilitates the dynamic QoS functions of negotiation, monitoring and adaptation. It is designed to be compatible with object-oriented distributed concepts such as interface and inheritance. It supports user-defined QoS categories. The dynamic matchmaking of offers with QoS requirements is also provided in QML with the help of checks that can investigate the QML specification dynamically. The three main abstraction mechanisms used in QML are: *Contract Type, Contract and Profile*. *Contract Type* explains the QoS aspects such as performance or reliability. A *Contract* is an instance of *Contract Type*, while the *Profile* is associates the *Contract* with interfaces, operations, operation arguments and operation results.

2.3.3.8 Web Service Management Language (WSML)

Web service Management Language WSML[107] is a language developed by HP Laboratories. It is an XML-based language which allows the definition of precise, formal, flexible and unambiguous specification of Service Level Agreements (SLAs) for Web services. In WSML an SLA is considered as a formal contract between two Web services which defines some guarantees offered by provider of Web service to the other parties such as consumer. The automated management of SLAs becomes easy with the use of precise and flexible specification of SLAs defined in WSML. The automation involves monitoring, enforcement, SLA optimization between Web services, and allowing SLA specification extensions in order to support the future additions into the SLA specification based on existing SLAs. The core elements of an SLA defined by WSML are: *Date Constraint*, *Parties* and *SLOs*. WSML does not support multiple service offerings into the specification for Web services. It also does not provide support for functional characteristics and access rights for Web services.

2.3.3.9 EXecutable Contracts (X-Contracts)

X-Contracts [89] offers the conversion of conventional contracts into electronically executable contracts by computers. The X-Contracts approach aims to reduce the ambiguities that are contained in human oriented contracts. The process includes the conversion of formal contracts into mathematical notations. The conventional contracts are represented with the help of Finite State Machines (FSMs). The FSM helps to

remove the ambiguities before the actual contract is converted into computer executable program. Initially the rights and obligations defined in conventional contracts are extracted then mapped to states, transitions and output functions accordingly. The middleware required for enactment of contracts is also described in X-Contracts. X-contracts support the monitoring and enforcement of rights and obligations for parties at run-time.

2.3.3.10 Business Contract Language (BCL)

Business Contract Language (BCL) [56] presents a formal system for defining contracts in terms of deontic logic related with concepts such as permissions, prohibitions and obligations. This logic supports reasoning about the violations of obligations in contracts. It is based on formalism for the representation of contrary-to-duty obligations which take place when other obligations are violated and penalties are applied in the contracts. This formalism is mapped to key policy contract specification, which is known as Business Contract Language (BCL). The BCL is a domain specific language, which supports abstractions used for expressing business contracts. BCL uses event pattern of specific style for expressing states of contract monitoring. Events are the core components of BCL. The BCL is also called as event-driven language. Event signifies the action of parties in the contract, temporal occurrences, change in the states, contract violations and conditions related with the contract execution.

2.4 Computational Services

Use of Computing services is growing in the form of Cloud computing, which is also known as Cloud infrastructure that helps providers and consumers to maximize its utilization. It also aims to reduce the costs incurred on the infrastructure and minimize the violations of Quality of Service that are usually defined in SLAs between service providers and service consumers. Cloud computing works as a utility provider of hosted hardware and software by delivering it as a service. Cloud computing literally tries to provide the provision of unlimited computing and storage resources on demand which is initially started with the use of small number of resources then increased to literally unlimited resources required.

According to the National Institute of Standards and Technology (NIST) [87], the Cloud computing is a paradigm which enables the access to a shared pool of configurable computing resources (i.e. Servers, networks, storage, applications and services). Cloud computing enables convenient, ubiquitous and on demand network access to the resources that can be provided and released rapidly with minimum efforts required by the service provider involvement.

The Cloud computing model is composed of five essential characteristics, three service models, and four deployment models. These five Cloud-computing characteristics described by NIST are given below :

On-demand self-service: the required computing resources can be provided automatically without human interaction required with each service provider.

Broad network access: it is can be used by heterogeneous types of clients such as laptops, workstations, mobile phones etc through standard network mechanisms.

Resource pooling: The computing recourses are shared and assigned to multiple consumers dynamically on demand. Providing only the abstract location (e.g. datacenter or country and state) of resources to consumers.

Rapid elasticity: the scale capacity of allocated resources can be increased or decreased on demand and also automatically without affecting the current resources in current use.

Measured service: provides fair provision of resources which are controlled and optimized automatically, also monitored and reported for utilized resources transparently to both the consumer and provider.

2.4.1 Cloud Service Models

A Cloud computing system has been categorized into three service models, which describe the type of the service offered by the Cloud provider. The three service models

e.g. Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) are described below [87]:

Infrastructure as a Service (IaaS): This model provides physical and virtual processing machines, networks, storage and other primary computing resources as a service where the consumers may run and deploy their own software such as applications and operating systems. In this model the management and control of the Cloud infrastructure used is not the responsibility of consumers.

Platform as Service (Paas): This model provides the capability to consumers for the deployment of consumer created or acquired applications such as operating systems, development platforms, programming languages, database platforms, libraries and tools etc on the infrastructure provided to them. The management and control of provided Cloud infrastructure is not the responsibility of the consumer, but they only have the control of configuration settings on their own deployed applications.

Software as a Service (SaaS): This model provides the capability to consumers to use the applications and data services of the providers. These applications can be accessed from various types of client devices. The underlying infrastructure provided is not managed or controlled by the consumer, only limited configuration settings allowed to the consumers. This model is known as the original Cloud service model.

2.4.2 Cloud Deployment Models

Cloud computing has four deployment models, which describe the way how a service running on a Cloud infrastructure is being deployed on the actual infrastructure. The deployment models are described below [87]:

Private Cloud: In this deployment model the infrastructure is provided for only use by one organization which may have more consumers in the form of business units. It can be operated and managed by the same organization or third party or combination of both within the on or off premises.

Community Cloud: In this deployment model the infrastructure is provided for only use by a particular community of consumers belonging to the organizations that have common concerns such as any particular mission, policy, security requirements and compliance consideration. It can be operated and managed by one or more organizations within the same community, or third party or combination of both within the on or off premises.

Public Cloud: In this deployment model the Cloud infrastructure is provided to general public for open use. It can be operated, managed, and owned by any government organization, academic or business or combination of them within the premises of the Cloud provider.

Hybrid Cloud: In this deployment model the Cloud infrastructure is provided in the combination of two or more different Cloud infrastructures such as public, private or community working independently, but adhering to the same proprietary technology or standard which involves the application and data portability such as use of Cloud bursting for balancing the load between different Clouds.

2.4.3 Computational Service Problems

In service oriented architecture (SOA), provision of computing as service (Cloud computing) gives new features added to the normal Web service interfaces. The Web services are described by WSDL, however, in order to publish the Cloud service description on the internet, the standard service description language for Cloud computing has not been standardized. The evaluation measurement methods for Cloud computing as compared to Web services require more QoS parameters due to dynamic nature and high complexity of the Cloud infrastructure. Service consumption in Cloud computing is also a challenge due to the scalable and dynamic nature of the Cloud paradigm. The SLAs for provision of computational services in the Cloud computing model needs more attention to be standardized.

2.4.4 Cloud Computing Service Providers

There are a number of Cloud computing service providers including Amazon, Google, Windows Azure and HP. The Cloud services that explicitly define SLAs are shown in Table 2-22, Table 2-23, Table 2-24 and Table 2-25. A full list is given in Appendix-A. In the Table 2-22 to Table 2-25, the columns from left to right describe the name of service, type of service and its category related to IaaS, PaaS or SaaS.

Table 2-22: Cloud Computing Services from Amazon with SLAs

Name of Service	**Type of Service**	**IaaS/PaaS/SaaS**
Amazon EC2	Compute	IaaS
Amazon S3 (Simple Storage Service)	Storage	IaaS
Amazon RDS (Relational Database Service)	Database	PaaS
Amazon Route 53	Networking	IaaS

Table 2-23: Cloud Computing Services from Google with SLAs

Name of Service	Type of Service	IaaS/PaaS/SaaS
Compute Engine	Compute	IaaS
App Engine	Compute	IaaS
Cloud Storage	Storage	IaaS
Cloud SQL	Storage	PaaS
Cloud Datastore	Storage	PaaS
BigQuery	Big Data	PaaS
Prediction API	Services	SaaS

Table 2-24: Cloud Computing Services from Windows Azure with SLAs

Name of Service	**Type of Service**	**IaaS/PaaS/SaaS**
Virtual Machines	Compute	IaaS
Web Sites	Compute	IaaS
Mobile Services	Compute	IaaS
Cloud Services	Compute	IaaS
Storage	Data Services	PaaS

SQL Database	Data Services	PaaS
HDInsight	Data Services	PaaS
Cache	Data Services	PaaS
Backup	Data Services	PaaS
Recovery Manager/ Recovery Services/ Hyper-V Recovery Manager	Data Services	PaaS
Media Services	App Services	SaaS
Service Bus	App Services	SaaS
Notification Hubs	App Services	SaaS
Scheduler	App Services	SaaS
Automation	App Services	SaaS
BizTalk Services	App Services	SaaS
Visual Studio Online	App Services	SaaS
Active Directory	App Services	SaaS
Multi-Factor Authentication	App Services	SaaS
Express Network/ Express Route	Network	IaaS
Virtual Network	Network	IaaS
Traffic Manager	Network	IaaS
CDN	Network	IaaS

Table 2-25: Cloud Computing Services from HP with SLAs

Name of Service	**Type of Service**	**IaaS/PaaS/SaaS**
HP Cloud Compute	Compute	IaaS
HP Cloud Block Storage	Compute	IaaS
HP Cloud Object Storage	Storage	IaaS
HP Cloud CDN Bandwidth	Network	IaaS
HP Cloud DNS	Network	IaaS

The Table 2-26 shows the use of refined SLAs elements taken from Table 2-6. The implicit elements are those elements which are mandatory elements of an SLA which exist in any SLA by default. The explicit list of elements are those elements which are deliberately mentioned into the SLAs by the service providers. The recommended elements by this thesis are the additional list of SLA elements suggested in order to well

document the structure of SLAs for efficient management and monitoring purpose for Cloud service SLAs.

Table 2-26: Implicit/Explicit and Recommended use of SLA elements

Refined SLA Elements Segments	SLA Elements	Implicitly Defined	Explicitly Defined	Recommended by This Thesis
Name	Agreement Name	X		
	Agreement Template			X
	Agreement Terms		X	
Purpose	Purpose	X		
Validity Period	Validity Period		X	
Parties	Service Provider	X		
	Service Consumer	X		
	Third Party			X
Party Role	Signatory Party	X		
	Supporting Party	X		
	Agreement Initiator	X		
	Agreement Responder	X		
Service Terms	Service Description Terms		X	
	Service Properties			X
	SLA Parameter			X
	Metric			X
	Measurement Directive			X
	Function			X
	Any Attribute			X
Guarantee Terms	Obligations			X
	Service Scope			X
	Service Level Objective		X	
	Action Guarantee		X	
	Penalties		X	
	Optional Services			X
	Restrictions			X
	Exclusions		X	

2.4.5 Cloud Computing Monitoring Tools

In order to get the values for QoS Term metrics for the computational Cloud services, there is a need of monitoring tools. Some providers provide these tools, while some third party venders also support the functionality to monitor the services of any service providers. Table 2-27 lists out the monitoring tools from most popular Cloud service providers that can be used to calculate the QoS for composite service plans generated from the proposed framework of this thesis.

Table 2-27: List of Cloud Service Monitoring Tools

S.No	Monitoring Tool Name	Features	Provider /Source
1	Amazon CloudWatch	Amazon CloudWatch can monitor AWS resources such as Amazon EC2 instances, Amazon DynamoDB tables, and Amazon RDS DB instances, as well as custom metrics generated by your applications and services, and any log files your applications generate.	[1]
2	Google Cloud Monitoring	Review performance metrics and logs for Google Cloud Platform services, VM instances, and common open source components.	[8]
3	Azure Management Portal	Set the level of monitoring to minimal and verbose for each service role, and can customize the monitoring displays. choose the metrics you want to monitor and you can choose which metrics to plot in metrics charts.	[4]
4	HP Cloud Monitoring	HP Cloud Monitoring is a managed service that monitors infrastructure, platform, and custom metrics to help ensure that workloads are operating at optimum levels.	[10]

2.5 Quality of Service

Quality of Service (QoS) is a key element for distinguishing between the services from different providers. The functional requirements describe the function of a software system or its components. A function is defined as a number of inputs, the behavior, and outputs. The QoS term is used for expressing non-functional requirements. The QoS describes how well the system performs its operations. The Business-to-Consumer

(B2C), Business-to-Business (B2B) and Consumer-to-Consumer (C2C) are the main business and consumer relationships, which are defined in Table 2-28. The Business-to-Business (B2B) e-commerce is highly attracted towards the use of Web services to solve the distributed computing challenges. The interaction with new business partners in the global business environment has become a critical issue for organizations. Selecting the best service out of those services that have similar functionality but different quality levels is a challenge. The selection mechanism needs to use some kind of intelligent decision-making systems. Sometimes it is not easy to measure the Quality of a Web service because the non-functional characteristics defined by QoS of a Web service can become wrong due to the false projection and fake advertisement from service providers. Therefore, it is a big challenge to calculate the true reputation of service providers [109].

QoS can be determined by two ways, the first is to know the QoS from the trust and reputation of the service providers that is not defined in the SLAs and the other is to determine the QoS from SLAs defined for the services. The QoS determined with the help of trust and reputations can include the QoS Terms from Table 2-32 which explains the factors affecting the quality of service providers, and QoS Terms from Table 2-30 which explains the QoS Terms from the classification of Trust. The Qos Terms defined in SLAs are the technical characteristics of the services, which are shown in Table 2-34, Table 2-35 and Table 2-36 and any domain specific terms are shown in Table 2-37.

Table 2-28 : Business and Consumer relationships in electronic Commerce

Business-to-Consumer (B2C)	The Business to Consumer (B2C) is used when Businesses sell their products or services to consumers.
Business-to-Business (B2B)	The Businesses sell their products or services to other Businesses.
Consumer-to-Consumer (C2C)	The ordinary consumers sell their products or services to other consumers with the help of electronically supported third Party Platform.

2.5.1 QoS from Trust and Reputation

Trust is defined as the extent to which one party wants to depend on somebody or something for a given situation with a consideration of relative security, despite the possibility of negative consequences [70]. Trust is considered as the personalized and subjective reflection of an individual's opinion. Trust can have two forms: one is the primary trust which specifies the trust built on direct personal relationship or observation, while the other is secondary trust about an entity which refers to the observation or relationship created from some other individual or organization [51]. Reputation is defined as the public's opinion about the character or standing of an entity (such as reliability, capability or honesty) which could be a person, an agent, an organization, a product or service [123]. A Trust and Reputation System (TRS) computes and discloses the Reputation score for an entity within the scope of a particular domain. The TRS computes the required scores of a reputation based on the opinions collected from public that hold or use the entities. The opinions are represented in the form of ratings or ranking score about an entity that is computed using some calculation methods. The score is stored dynamically on some central or distributed locations of TRS systems. Based on the Reputation score, people may make decisions whether or not to buy the services or goods they are looking for. The rating or ranking defines the relationship between the entities that are identified with more, less or equal degrees of accreditation.

2.5.1.1 Online Reputation Systems

Online Reputation Systems (ORS) accumulate individual opinions about the entities such as a person, an agent, an organization, a product or a service in the form of Reputation score or information, then they process that score according to some algorithm and disseminate it online for public, so that other people can use the Reputation information as a reference to take their decisions about the products or services accordingly [102][45].

2.5.1.2 Classification of Reputation Systems

The Classification of Reputation Systems by Ling Liu [81] is explained in Table 2-29. Reputation Systems are classified according to Network Architecture, Information Source, e-Business Mode and Functions.

Table 2-29 : The Classification of Reputation Systems

Network architecture: Reputation systems can be divided into two types which are based on the information storage location [70]: **1. Centralized Reputation Systems:** They rely on a central entity for gathering, computing and disseminating reputation information. Centralized reputation systems are widely used in the areas of: Consumer-to-Consumer (C2C) markets, online retailers, shopping comparison sites and information communities. **2. Distributed Reputation Systems:** They rely on decentralized solutions where every peer stores information about the other peer for interaction. Reputation information is disseminated on demand between peers. Distributed systems are mainly used within Peer-to-Peer systems.
Information Source: Reputation Systems are divided into Explicit and Implicit Systems based on Information Source [37]. **1. Explicit Source:** Explicit sources voluntarily write reviews and provide the ratings about the entities. **2. Implicit Source:** In the Implicit Source the information is derived from users' activities. Example: the number of sales made by an entity is directly proportional to the increase in the ranking of the entity.
E-Business Mode: Reputation Systems classified based on e-Business model are divided into Bidirectional and Unidirectional Systems [37][60]. **1. Bidirectional Systems:** The Bidirectional Systems are mostly used in Consumer-to-Consumer (C2C) and Peer-to-Peer (P2P) web sites allowing the users to rate or rank each other. **2. Unidirectional Systems:** In Unidirectional Systems, only the consumers give the ratings or write the reviews on products or services, mostly used by Business-to-Consumer (B2C) Companies.
Functions: Classification of Reputation Systems according to Function is divided

into three types [81]:

1. Trust Building: Most C2C marketplaces and Job Centers use reputation systems for building trust among the buyers and sellers.

2. Reducing Information Asymmetry: online retailers, price comparison sites and review centers depend on reputation systems to reduce information asymmetry.

3. Information Filtering: Information centers and online forums adopt reputation systems to filter information.

2.5.1.3 Classification of Trust

The classification of Trust by Grandison & Sloman [57] is defined in Table 2-30. The classification of Trust is based on Access Trust, Provision Trust, Certification of Trustee, Delegation Trust and Infrastructure Trust.

Table 2-30 : The Classification of Trust

Access to Trustor's Resources (Access Trust): A Trustor needs to trust a trustee to use the resources offered by the trustee, or controls provided by trustee in the form of a software execution environment or an application service.
Provision of Service by the Trustee (Provision Trust): The trustor needs to trust the trustee for the service being offered and assumes that the trustee will not access the trustor's resources. Application Service Providers (ASPs) and service bureaus are major examples of entities that require service provision trust to be created.
Certification of Trustees: The certification of trust is based on a certificate of the trustworthiness of a trustee ensured by a third party. The type of certificate presented by the trustee to the trustor are based on a certain criteria predefined in the certification. The authentication of the identity on internet applications is approved through the certificates.
Delegation Trust: A trustor trusts on Trustee to make the decisions on his behalf related to any resource or service that the Trustor wants to use or avail. This may also be considered as the trust making service in the form of service provision from trustee.

Infrastructure Trust: The infrastructure trust is related with the basic infrastructure used to provide the services to trustor, and must be trusted in order to get the services from trustee. The infrastructure includes the hardware and application softwares certified by third parties.

2.5.1.4 Types of Attacks on Trust and Reputation Systems

The types of attacks described by Audun [69] are shown in Table 2-31.

Table 2-31 : Types of Attacks on Trust and Reputation Systems

Playbooks: In a playbook, a series of actions are carried out which will increase the fitness or profit of a participant based on certain criteria. In this way someone acts honestly and maintains the quality of their services for specific time to gain a high score reputation, then after getting the benefit of timely created high reputation score they suddenly provide low quality services at a low production cost.
Unfair ratings: In this kind of attack the raters provide the fake opinion about the entities. This is unethical and it is also very difficult to identify when this attack is made, because the community does not know exactly who has given the genuine or fake opinion.
Discrimination: A service provider may provide a high quality service to one group of parties, while another group of parties is being served with low Quality Services. This behaviour can have very different effects on the service entity's reputation score depending on the specific TRS being used.
Collusion: In collusion agents create a situation for a kind of behaviour which can lead to running the playbooks, and result in unfair recommendations or discrimination. Collusion is a kind of informal agreement which is used against the competitors to achieve the company's goals such as getting more customers by using a trick of price reduction in services or products without official announcement.
Proliferation: Offering the same service through many different channels, or same product being advertised by multiple representatives. It is unethical when the same service is being presented and pretended to be different and independent services.
Reputation Lag Exploitation: The service entity's reputation score can be affected

if there is a time lag between the instance of service provision and generation of its ratings. It is possible that during the use of time lag, a large number of services can be provided with low Quality, before the Rating score is affected due to the provision of low quality services.
Re-entry: When an agent or entity with low score leaves the community or reputation system and then again participates in the community or reputation system with a different identity. They need to start from fresh and possibly representing the same entity with different identities at the same time which is unethical.
Value Imbalance Exploitation: Receiving reputation with large number of high quality services of low cost value and then selling high value cost services with low quality is misleading and it will result in high profit to service providers and will not reduce overall score of provider Reputation significantly which is unethical. The value of service should be given some weight in reputation calculation accordingly in order to solve this weakness.
Sybil attack: The Sybil attack was named after the subject of a book Sybil, a case study of a woman with multiple personality disorder. In a Sybil attack, a substantial malicious user get hold of multiple fake identities for itself and pretends to be multiple distinct nodes (Sybil nodes) in the system[90]. A single entity that creates various fake identities (pseudonyms) within a Trust and Reputation System (TRS) for a particular domain, and provides the multiple ratings on the same Service object.

2.5.1.5 Factors Affecting the Reputation of Service Providers

The factors that influence the reputation of service providers defined by Sha [109] are shown in Table 2-32.

Table 2-32 : Factors Affecting the Reputation of Service Providers

Life span: The length of time for which a particular Web service remains live online and functions properly in the market by attracting their target customers.
Financial status: In order to make a business strong and its maintenance with wise financial decisions is difficult task. But if there is more cash flow in the accounts of business, it will make the business healthier financially, and make the business

organization more trustworthy. The progress of the business also depends on the financial status of the company.
Branches: A business is expanded by opening its more branches locally and globally. It helps spreading the business to diverse locations, bringing their products and services closer to customers more easily and effectively. The increased number of branches of a company or business will help the business expand the size of the market for company's products and services by attracting more number of Customers.
Employees of organization: The most important assets of an organization are the human resources. People with high qualification, experience and mandatory skills can only bring the positive and creative contributions to the organization. Therefore selection of correct employees for the organization is most crucial.
No. of services: A company with more number of services maintaining the good Customer service and Quality of Service or goods will have good reputation. If a customer is satisfied then the client base can be increased rapidly, but if someone is dissatisfied with service or good, then it is a danger and reputation of company can be destroyed. Therefore increased number of satisfying services is good for reputation of company and more chances of survival for the company.
Brand value: Brand value is the additional income to a company occuring because of its brand name. People are willing to pay more for a particular brand as compared to ordinary products with less popularity. The services or goods of a company can gain a brand value once they become a benchmark and their experiences and performances become obvious to the consumers.
Success rate: The achievement of the desired goals of a company is called the success. The success rate of the company is directly influenced with the customer satisfaction. The company's growth also depends on the measure of success rate of planned goals.
Advertising: Advertising is a method of communicating with intended audience or customers of a company for encouraging them to avail the services or products with confidence. Advertisement is most of the time a paid task and transmitted by various media sources. The strong advertisement assures the company shareholders and also to the customers about the success of company.

The Table 2-33 shows the list of QoS Terms collected together that are not defined within SLAs, but are the important QoS characteristics used as QoS metrics that can be included for service providers trust and reputation evaluation purpose.

Table 2-33: QoS Terms defined without the SLA Parameters

Category of Terms	List of Terms
Service Provider Factors	1. Life Span 2. Financial Status 3. Branches 4. Employees Of Organization 5. No Of Services 6. Brand Value 7. Success Rate 8. Advertising
Trust	1. Access Trust 2. Provision Trust 3. Certification of Trust 4. Delegation Trust 5. Infrastructure Trust

2.5.2 QoS from Service Level Agreements

Various Quality of Service terms have been used by different researchers in different kinds of domains as metrics for QoS calculation. The W3C working group [80] summarized the definition of terms for Quality of Service and its metrics according to Web service requirements. There are two groups of terms i.e. Performance and Security metrics defined by W3C, while the other terms have been classified under the category of dependability metrics [20] [64]. Since domain specific Web services are entirely different in nature, it is impractical to describe all QoS metrics for all of the domains in a single model. Therefore, the fourth group, “application-specific metrics”, is also created for the metrics that are specific to a particular domain [123]. The tables (Table 2-34, Table 2-35, Table 2-36 and Table 2-37) list out all four groups separately.

Table 2-34 : QoS Terms from SLAs on Performance metrics [80]

Attribute No.	Attribute Name
1	Throughput
2	Response Time
3	Latency
4	Execution Time
5	Transaction Time

Table 2-35 : QoS Terms from SLAs on Security metrics [80]

Attribute No.	Attribute Name
1	Authentication
2	Authorization
3	Accountability
4	Confidentiality
5	Traceability and Auditability
6	Non-Repudiation
7	Encryption

Table 2-36 : QoS Terms from SLAs on Dependability [123]

Attribute No.	Attribute Name
1	Availability
2	Accessibility
3	Accuracy
4	Reliability
5	Capacity
6	Scalability
7	Exception Handling (Stability)
8	Robustness (Flexibility)
9	Integrity (Data and Transaction)

Table 2-37: QoS Terms from SLAs on Domain Specific metrics [123]

Attribute No.	Attribute Name
1	Any Attribute

2.5.2.1 QoS Terms Definitions from SLAs

The QoS Terms based on Performance metrics are explained in Table 2-38. The Performance of a Web service is described as how fast a service request can be completed. It is a measure of speed in completing a service request [80].

Table 2-38: QoS Terms from SLAs on Performance metrics Explained

Throughput: Throughput represents the number of Web service requests served in a given time interval [100][114].
Response Time: Response represents the time required to complete a Web service request [100][114].
Latency: Latency represents the round-trip delay (RTD) between sending a request and receiving the response [100][114].
Execution Time: The execution represents the time taken by a Web service to process its sequence of activities [100][114].
Transaction Time: The Transaction time describes the time that passes while the Web service is completing one complete transaction. This transaction time may depend on the definition of Web service Transaction [100][114].

The QoS Terms based on Dependability metrics are explained in Table 2-39. The Dependability is defined as the Probability that a computer or other system will perform its intended functions in its specified environment without significant degradation [5].

Table 2-39: QoS Terms from SLAs on Dependability metrics Explained

Reliability: Reliability represents the ability of a Web service to perform its required functions under stated conditions for a specified time interval [54]. It is the overall measure of the ability of a Web service to maintain its service quality. Reliability also assures the delivery of message being transferred and received by service Requesters and service providers [114].
Scalability: The Scalability represents the capability of increasing the computing capacity of service provider's Computer System and System's ability to process more

users' requests, operations or transactions in a given time interval [100].
Capacity: The Capacity is the limit of the number of simultaneous requests which should be provided with guaranteed performance [100].
Robustness: Robustness represents the degree to which a Web service can function correctly even in the presence of invalid, incomplete or conflicting inputs [100].
Exception Handling: Since it is not possible for the service designer to specify all the possible outcomes and alternatives (especially with various special cases and unanticipated possibilities), exceptions should be handled properly [100].
Accuracy: Accuracy here is defined as the error rate generated by the Web service [100].
Integrity: Integrity for Web services should be provided so that a system or component can prevent unauthorized access to, or modification of, computer programs or data. There can be two types of integrity: data integrity and transactional integrity. Data integrity defines whether the transferred data is modified in transit. Transactional integrity refers to a procedure or set of procedures, which is guaranteed to preserve database integrity in a transaction [114].
Accessibility: Accessibility here represents whether the Web service is capable of serving the client's requests [114].
Availability: The degree to which a system or component is operational and accessible when required for use [68]. The service should be available immediately when it is invoked.
Interoperability: Web services should be interoperable between the different development environments used to implement services so that developers using those services do not have to think about which programming language or operating system the services are hosted on [114].

The QoS Terms based on Security metrics are explained in Table 2-40. With the increase in the use of Web services which are delivered over the public Internet, there is a growing concern about security. The Web service provider may apply different approaches and levels of providing security policy depending on the service requestor [80].

Table 2-40: QoS Terms from SLAs on Security metrics Explained

Authentication: The Authentication describes how the service authenticates principals (users or other services) who can access the service and data [100].
Authorization: How the service authorizes principals so that only they can access the protected services [100].
Confidentiality: How the service treats the data, so that only authorized principals can access or modify the data [100].
Accountability: The supplier can be hold accountable for their services [29].
Traceability and Auditability: It should be possible to trace the history of a service when a request was serviced [100].
Data encryption: It describes how data should be encrypted [100].
Non-Repudiation: A user cannot deny requesting a service or data after the fact [100].

The QoS Terms related to a particular domain are explained in Table 2-41. The domain specific terms are specially defined due to specific attributes which need to be defined for particular needs.

Table 2-41: QoS Terms from SLAs on Domain Specific metrics Explained

Any Attribute: The application specific or domain specific metrics or attributes are defined according to domain needs that are specific to a particular domain [123].

The Table 2-42 shows the list of QoS Terms collected together as metrics that are explicitly defined within SLAs and they are the important measureable QoS characteristics used for determining the QoS ranking for service providers.

Table 2-42: QoS Terms defined within SLA Parameters

Category of Terms	**List of Terms**
Performance	1. Throughput 2. Response Time 3. Latency 4. Execution Time 5. Transaction Time
Security	1. Authentication

	2. Authorization 3. Accountability 4. Confidentiality 5. Traceability and Auditability 6. Non-Repudiation 7. Encryption
Dependability	1. Availability 2. Accessibility 3. Accuracy 4. Reliability 5. Capacity 6. Scalability 7. Exception Handling (Stability) 8. Robustness (Flexibility) 9. Integrity (Data and Transaction)
Domain Specific Terms	1. Any Attribute

2.5.3 QoS Terms defined within and without SLA Parameters

The QoS Terms defined without SLA Parameters from Table 2-33 and QoS Terms defined within SLA Parameters from Table 2-42, are combined and shown in Table 2-43.

Table 2-43: QoS Terms defined within and without SLA Parameters

Category of Terms	**QoS Terms**
Service Provider Factors	1. Life Span 2. Financial Status 3. Branches 4. Employees Of Organization 5. No Of Services 6. Brand Value 7. Success Rate 8. Advertising
Trust	9. Access Trust 10. Provision Trust 11. Certification of Trust 12. Delegation Trust 13. Infrastructure Trust
Performance	14. Throughput 15. Response Time 16. Latency 17. Execution Time 18. Transaction Time
Security	19. Authentication 20. Authorization 21. Accountability 22. Confidentiality 23. Traceability and Auditability 24. Non-Repudiation 25. Encryption

Dependability	26. Availability 27. Accessibility 28. Accuracy 29. Reliability 30. Capacity 31. Scalability 32. Exception Handling (Stability) 33. Robustness (Flexibility) 34. Integrity (Data and Transaction)
Domain Specific Terms	35. Any Attribute

2.5.4 QoS Selection Techniques

In the environment of Cloud computing services, the assessment of QoS becomes very hard due to the different types of QoS metric terms used. There is a range of service provisioning techniques used for QoS calculation, while the major approaches are: (a) Algorithmic, (b) Multi-Criteria Decision Making (MCDM), (c) SLA and Policy Based Brokering and (d) Heuristic and holistic [126].

This thesis focuses on MCDM. Multiple-Criteria Decision Analysis (MCDA) [55] enables problems such as selection of services based on multi-criteria QoS terms to be dealt with. The MCDA is a field of operations research which is based on multiple criteria in decision-making environments. It helps in selecting the best alternative services among several choices. MCDA has different methods which are based on matrices such as evaluation matrix, decision matrix, payoff matrix or evaluation table [101].

MCDA has two main categories: Multi-Attribute Decision Making (MADM) and Multi-Objective Decision Making (MODM) [88]. In MADM the alternatives are predetermined from a set of multiple attributes and then a small subset is evaluated against it, while in MODM the alternatives are not pre-specified but are driven by optimizing the set of objective functions.

The most popular MCDA methods used in the state of the art [125] [53] are Analytic Hierarchy Process (AHP), Analytic Network Process (ANP), Technique for Order of Preferences by Similarity to Ideal Solution (TOPSIS), Elimination and Choice Expressing Reality (ELECTRE), Preference Ranking Organization Method of

Enrichment Evaluations (PROMETHEE), Decision-Making Trial and Evaluation Laboratory (DEMATEL), Grey Relational Analysis (GRA), VIKOR, Fuzzy sets, Goal Programming and Data Envelopment Analysis (DEA).

2.5.4.1 Analytic Hierarchy Process

The Analytic Hierarchy Process (AHP) [106] is a method of MCDA for analyzing and organizing complex decisions, derived from psychology and mathematics particularly for group decision making, it was proposed by Saaty in 1988. AHP supports both quantitative and qualitative type of criteria among the alternatives where the attributes are mostly dependent on each other. It works with pairwise comparison, using structured attributes into a hierarchical relationship from top level to the goal, where the sub criteria are connected to the upper level criteria. The process is started from leaf nodes to the top level within hierarchy tree. Different branches originated for each level have a corresponding influence or weight for each output level in the hierarchy. In the last step, the best suited alternative for each attributed is selected as final output [42].

2.5.4.2 Analytic Network Process

Analytic Network Process (ANP) [105] is an extended form of AHP proposed by Saaty in 1996. It is a detailed decision-making method proposed to solve the problem of feedback and dependence among the criteria. It uses hierarchical interrelationships between decision levels and attributes in unidirectional way. It incorporates the ratio scale measurements depending on pairwise comparisons to address the decision problem. ANP uses a supermatrix containing composite weights for handling the interdependence among the elements [104]. It has been used in many real world decision-making problems.

2.5.4.3 Technique for Order of Preferences by Similarity to Ideal Solution

The Technique for Order of Preferences by Similarity to Ideal Solution (TOPSIS) method was introduced by Hwang and Yoon in 1981 [65]. In the TOPSIS technique, initially the decision matrix is normalized with the help of vector normalization and then the anti-ideal and ideal solutions are determined using the normalized decision matrix. In this method, the alternatives are selected from the positive ideal solution

having shortest distance and the negative ideal solutions are selected from the farthest distanced alternatives [88].

The TOPSIS method identifies multiple criteria solutions from a finite set of alternatives. According to TOPSIS the best solution should have the shortest distance from the positive ideal solution and the farthest distance from the negative ideal solution [42]. This method uses an aggregating function which uses the distances from the negative-ideal point and positive ideal point without concerning their relative importance, while the reference point is still important for decision making and it should be closest possible to the ideal solution [101].

2.5.4.4 Elimination and Choice Expressing Reality

The Elimination and Choice Expressing Reality (ELECTRE) method was introduced by Roy and Vanderpooten in 1996 [103]. It has several versions including ELECTRE (I to IV, IS and TRI). The ELECTRE is based on two types of parameters: the veto thresholds and the importance coefficient [88]. This methods is computationally complex as compared to other methods, it has 10 steps in the simple form. The outranking relationships are determined by using the pairwise comparison between the alternatives. This methods is used to select the best alternative with maximum advantage and least variance in the function of various criteria Then the alternatives dominated by others are identified and eliminated by these relationships and hence by giving the smaller set of alternatives. This method deals with discrete criteria that are both qualitative and quantitative in nature, by offering the complete sequence of alternatives. The preference of the alternatives depends mostly on the criteria, concordance, discordance, discordance indices, graphs and threshold values of relationships. The ranking of alternatives is obtained by using the graphs in an iterative procedure [101].

2.5.4.5 Preference Ranking Organization Method of Enrichment Evaluations

The Preference Ranking Organization Method of Enrichment Evaluations (PROMETHEE) method was introduced by Brans and Vincke in 1985[26]. It is an

extended version of outranking method ELECTRE, but it is different at the pairwise comparison stage, but both methods identify the best possible alternative. PROMETHEE has additional features to consider the degree of better option and with the help of this information, it helps to identify the non-dominated or least dominated alternatives and eliminates the dominated alternatives. PROMETHEE is easier to use and less complex as compared to ELECTRE for ranking the alternatives [42].

2.5.4.6 Decision-Making Trial and Evaluation Laboratory

The Decision-Making Trial and Evaluation Laboratory (DEMATEL) method was proposed by Gabus and Fontela in 1973 [52] . In this method the factors are represented as the interrelationships among the criteria. The DEMATEL is a comprehensive method for creating structural model using the associations of complex factors. The numerical representations of power of influence is used within a system to organize the relationship between the elements within a system [30]. This method represents the strength of the influence by a number value. The relationships of the elements in the method are visualized contextually using matrices or diagraphs. This method has been effectively used in the variety of situations such as developing control systems, marketing strategies, solving safety problems and group decision-making [129].

2.5.4.7 Grey Relational Analysis

The Grey Relational Analysis (GRA) theory was proposed by Deng in 1982 [46]. This method uses a grey relational degree obtained depending on the changing alternatives into comparable sequence and identifying an ideal target sequence. The word “grey” represents a color that advises the quantity of the known information used in the control theory. This GRA system theory is mostly used to deal with incomplete, poor and uncertain information. The problems related with complex interrelationships between the variables and factors are solved by GRA, along with range of MADM problems. The results of GRA are based on original data, and its calculations are simple and straightforward [124].

2.5.4.8 VIKOR

VIKOR method was proposed by Opricovic in 2004 [92]. It is a ranking method for compromise evaluation and optimization in complex and dynamic processes in multi-

criteria decision-making. This method uses measures of closeness to the ideal solution for identifying the multi-criteria ranking index. The VIKOR method uses linear normalization, but the normalized values are not dependent on the evaluation unit of a criterion. The distance between the individual and ideal satisfaction is balanced by aggregating function in order to obtain the ideal solution. VIKOR is one of the efficient MCDA method which is used for sorting, ranking and also for the selection among the alternatives involving conflicts [16].

2.5.4.9 Fuzzy Sets

The Fuzzy sets were introduced by Zadeh in 1965 [130]. This method has been widely used to deal with the ambiguities involved in human judgment. This method properly helps to resolve the uncertainties found in the available information given for multiple criteria decision-making purpose. A Fuzzy Method helps to evaluate alternative criteria based on decision pool known as Fuzzy Associative Memory. In Fuzzy sets, Fuzzy terms are described by using the linguistic variables which are used to map with the numerical values. The Fuzzy unit intervals are used in decision making process in place of Boolean truth values used in conventional sets [67].

2.5.4.10 Goal Programming

The goal programming was first employed by Charnes et al in 1955 [33], it is a Multi-Objective Decision Making (MODM) tool. Goal Programming is and extended version of linear programming which is usually used to solve the problems having multiple and conflicting objects. The multiple conflicting objective measures are handled by the optimization used in this technique which is achieved by minimizing the irrelevant information. The logic of optimization is combined with mathematical programming in multi-criteria decision making in order to fulfill several objectives [97].

2.5.4.11 Data Envelopment Analysis

The Data Envelopment Analysis (DEA) was introduced by Charnes et al. in 1978 [34]. Initially the DEA was designed for performance measurement but later on it became a comprehensive theoretical framework. The DEA is used for evaluating the competence of an observation relative to a set of similar observation, and it is a mathematical programming technique [35]. DEA focuses on measuring the effectiveness of multiple

decision making units, in an environment with multiple outputs and inputs. DEA finds the efficient mixture of multi inputs and multi outputs of the problem. DEA is used in industries to measure the impact of multi-criteria decision making systems [88].

2.5.5 Limitations/Drawbacks in QoS Selection Techniques

The Table 2-44 shows the limitations and drawbacks found in different MCDA techniques for QoS selection [53][62] [122] [115] [108]. It is clear from Table 2-44, that the Fuzzy sets are worth investigation because they are flexible and manageable due to the use of linguistic variables used in the Fuzzy Inference process and they can be easily converted to numerical values. However, the other MCDA approaches have significant difficulties and limitations.

Table 2-44: Limitations/Drawbacks in QoS Selection Techniques

Approach	**Limitation/Drawback**
Analytic Hierarchy process (AHP)	• The number of pairwise comparisons can become too many which can lead to lengthy task if the number of levels are increased in the hierarchy • Difficult to distinguish the scale due to limitation of Nine point scale • Solution to the linear equations do not always exist • Too much computation required even for small problem • Supports only triangular Fuzzy numbers • The number of pair comparisons increase if the number of levels in the hierarchy increases, and hence it takes more efforts and time • Difficult to use when the number of criteria or alternatives is high for example more than 7 • Adding new criterion or alternative is difficult
Analytic Network Process (ANP)	• Method is Time consuming • Uncertainty is not supported • Decision made is hard to prove • It is difficult to provide accurate network structure among criteria • The different network structures can produce different results

Technique for Order of Preferences by Similarity to Ideal Solution (TOPSIS)	• Easy to implement but can give unreliable results. • Does not consider the uncertainty in weightings. • Deterministic in its standard form
Elimination and Choice Expressing Reality (ELECTRE)	• Difficult to understand how to find the concordance and discordance matrices • Thresholds are calculated based on metrics, therefore it is difficult to convert subjectivity opinion into thresholds value. • Process and results are difficult to understand without professional knowledge of the method. • Is time consuming
Preference Ranking Organization Method of Enrichment Evaluations (PROMETHEE)	• Result is affected when new alternative is added • Does not give support to structure a decision problem • Becomes complicated when various criteria and options become available • The solution is not straight forward
Decision-Making Trial and Evaluation Laboratory (DEMATEL)	• The direct-relation matrix must be created by expert grading or questionnaire
Grey Relational Analysis (GRA)	• Does not always give the optimal solution
VIKOR	• In case of conflicting situations, a decision maker may take imprecise or ambiguous data and the performance rating is calculated as values
Fuzzy Sets	• Uses Linguistic variables for describing Fuzzy terms that are then mapped to numerical variables
Goal Programming	• Setup of appropriate weights require a lot of work.
Data Envelopment Analysis (DEA)	• The error caused by measurement can cause significant problems • The statistical tests are not applicable • The absolute efficiency cannot be measured by this method • Large problems require a lot of working

2.5.6 Advantages of using Fuzzy Inference System

According to state of the art, Fuzzy Inference System method is the most frequently used among all other MCDM techniques [88] [125]. The proposed framework

SLAAgent (Figure 5-2) in this thesis uses Fuzzy Inference System for QoS calculation. The core advantages of FIS used in the SLAAgent are listed in Table 2-45.

Table 2-45: Advantages of Fuzzy Inference System

S.No	Advantage
1	Can support decision making individually as well as in the groups.
2	Allows weights to be setup for any Inputs if required.
3	Works well for incomplete or imprecise data given for problems.
4	Suitable for both quantitative and qualitative data used in decision making.
5	Can be used for wide range of areas involving decision-making based on different number of inputs.
6	It is fast and does not involve complex calculations.
7	Requires no comparisons, only min or max operators within the rules.
8	Easy to implement.
9	Easy to use in any domain of interest involving multi criteria decision-making.
10	Can work independent of relation between Inputs and Output.
11	Can work with different units of inputs and outputs.

2.5.7 Discussion of different approaches using Fuzzy Inference System

Cloud computing is recognized as the most popular paradigm of computing resources on demand which supports an incredible large amount of computational power and storage. Due to a huge number of Cloud service providers, it becomes very difficult for the consumers to select the most suitable service provider. The trust and reputation systems help to select the most suitable services among the pool of similar services from different service providers. At the same time, it is also difficult to choose the most suitable QoS calculation technique for developing trust and reputation system.

QoS can be calculated based on various QoS metric terms employed by any service provisioning technique. Under the Multi-Criteria Decision Making (MCDM) technique, several approaches have been used in a variety of applications for QoS calculation [125] [53].

Fuzzy Logic is the most popular and frequently used MCDM technique. Several approaches including [99] [49] [44] [98] [19] and [21] have used the Fuzzy Inference System for service selection method.

Qu and Buyya [99] proposed a Cloud trust evaluation system using hierarchical Fuzzy Inference System for service selection. The approach evaluates trust of Clouds according to a user's Fuzzy Quality of Service requirements and services' dynamic performances to facilitate the service selection. The approach has employed Fuzzy membership functions to capture the users' subjective preferences and requirements for different QoS terms and then it uses a hierarchical Fuzzy Inference System to derive the trust level. The approach tries to ease the IaaS selection process for both expert and inexperienced users by modeling their unclear requirements and vague preferences with the use of linguistics descriptors.

Frey et al. [49] proposed an extended QoS provisioning architecture for Cloud QoS scaling using Fuzzy Logic. The approach has tried to show that with additional imprecise information with the help of additional input parameters using Fuzzy Logic, the up and down scaling mechanism of a Cloud service can be optimized. They have also tried to minimize the violation of SLA elements.

Dastjerdi and Buyya [44] proposed a compatibility aware Cloud service composition under Fuzzy preferences of users. This approach presents a framework and algorithms which simplify Cloud service composition for unskilled users. The approach is ontology based that analyses Cloud services compatibility by applying reasoning on expert knowledge. In order to minimize the efforts of users in expressing their preferences, combination of Fuzzy Logic and evolutionary algorithms is applied for composition optimization. The approach has tried to give the users more control by using linguistic terms for expressing the user preferences as compared to conventional assignment of

exact weights for their preferences. The approach also aims to provide the support of semantic interoperability, ease of service selection to non-experts, monitoring of SLAs and support of negotiation strategy.

Prasath et al. [98] proposed a model for web service selection using Fuzzy quality of protection. This approach aims to solve the selection of secure web services in global manner. This approach considers quality of protection parameters such as tampering, spoofing, information disclosure, reputation, denial of service and elevation of privileges as input and risk rating as output in fuzzy inference method.

Avila and Djemame [19] proposed Fuzzy Logic based QoS optimization mechanism for Service composition. This is an adaptation approach that implements self-optimization based on Fuzzy Logic. The proposed optimization model performs service selection based on the analysis of real and historical QoS data, collected at the execution of composite services. The use of Fuzzy Inference Systems enables the evaluation of the measured QoS values, helps deciding whether adaptation is needed or not, and how to perform service selection. Fuzzy logic is used as decision making tool to determine the need of adaptation in the context of service compositions.

Baliyan and Kumar [21] proposed an approach for quality assessment of software as a service on Cloud using Fuzzy Logic. It uses quality attributes as inputs to the Fuzzy Inference method. The proposed model has used the Fuzzy toolbox of MATLAB. This model has demonstrated various possible combinations of values of quality attributes. The model enables users to choose a service according to a tailored definition of quality comprising quality characteristics desired on demand.

2.6 Web Service Composition

Composition of services is the most demanding procedure needed to support Business-to-Business and enterprise applications integration for utilizing multiple services in a distributed environment [22][112]. Web service composition is a way of integrating different Web services for creating a high- level business process. It helps to combine the atomic services to provide a joint functionality that cannot be received from single service at design time. The outcome of Web service composition is the new

functionality achieved by reusing the existing Web service components available that are not able to complete a required task alone.

2.6.1 Web Service Composition Approaches

The main advantage of Web services is the reusability and creating the new services based on the existing services. Service composition methods define how to compose those services. They also describe the order of the services invoked as well as the conditions related with combined execution of the services.

Web service composition can be divided into static and dynamic forms [66]. The static Web service composition as the name implies is fixed and cannot be altered after execution of the services. The dynamic Web service composition is modifiable whenever the requirements are changed for Web service at runtime. Hence, the formation of dynamic Web service composition as compared to static Web service composition requires more efforts to achieve the composite tasks.

The composition approaches are divided into two types: Syntactic and Semantic Web service composition [22].

2.6.1.1 Syntactic Web Service Composition

Two main approaches used for syntactic Web service composition (static service composition) which are orchestration and choreography explained below:

A. Orchestration

In orchestration the Web services are controlled by a single endpoint central process, which coordinates the execution of various operations of Web services involved in the process. The participating Web services do not need to know their involvement in the composition process. Only the central coordinator of the process needs to hold this information [71]. A main approach of orchestration is BPEL4WS which is explained below:

Business Process Language for Web Services (BPEL4WS):

The Business Process Language for Web Services (BPEL4WS) [17] is a formal specification used to define the business processes and interaction protocols. It gives an

extension of business transactions to the Web service interaction model. BPEL4WS gives the interoperable integration model which supports the automated process integration for Business-to-Business and intra-corporate. BPEL is an XML- based language for orchestration, standardized by Organization for the Advancement of Structured Information Standards (OASIS) [12]. BPEL has process-oriented type of service composition in the form of business process or workflow [47]. BPEL is considered as the behavioral extension of WSDL with workflow based approach.

A BPEL business process has three main entities [41] 1) Partners: Every partner uses the WSDL description of Web service for activities. A partner link explains which activity is connected to a particular Web service provider, while a Web service provider is seen as a port of particular port type. 2) Variables: used to store messages and process states are stored into variables. 3) Basic Activities: every activity has attributes and elements. The activities define the business logic. The activities can be invoking Web service, value assignment to a variable, structuring and executing activities in parallel or in sequence.

B. Choreography

Choreography is not organized by the central coordinator. Each Web service in the process has to know when and where to execute. It is based on collaboration used to exchange messages in public business processes. All participating Web services of the business process must be aware of business process, execution of operations, messages being exchanged and the exact timings of invocation of operations [71]. The main approach of choreography is WS-CD which is explained below:

The Web Services Choreography Description Language WS-CDL:

The Web Services Choreography Description Language WS-CDL [72] is an XML-based language for Peer-to-Peer collaborations. The WS-CD Specification aims to compose any type of participants independent of platform support and programming model. Its main component is Interaction activity which exchanges the information between parties major focus on receiver of the information [22]. WS-CDL has three main parts: participants, information and channel for exchanging the information. Exception handling also supported. Messages exchanged between participants are

carried out by variables and tokens and their types defined by XML schema and WSDL. Channels define how and where the message are exchanged. Synchronization between the activities is also supported.

2.6.1.2 Semantic Web Service Composition

Semantic Web service composition is a kind of dynamic service composition, the popular approaches are Semantic Markup for Web Services (OWL-S) and Web Service Modeling Ontology (WSMO) which are explained below:

1. Semantic Markup for Web Services (OWL-S)

The Ontology Web Language Semantics (OWL-S) [86] is an approach used to define the ontology for Semantic Markup of Web services. The OWL-S aims to support the automation of Web service discovery, composition, invocation, interoperation, execution and monitoring with the help of Semantic Descriptions of Web services. The OWL-S (Ontology Web Language for services) is defined in four main elements [22]. The first element: *Service* concept, which represents an organizational point of reference for exposing the Web services. *Service* instance is used to declare every Web service. Service is linked by other three elements using properties such as *Presents*, *Describedby* and *Supports*. The second element: *Service Profile* defines a high-level description of Web service, its functionality and non-functional characteristics, and is used to discover the Web service using semantic descriptions. The third element: Service Model defines how a Web service gets its functionality such as detail about the business process involved in process model. The fourth element: *Service Grounding* defines how to access and use the Web service and how Web service consumers can invoke the Web service from different locations.

2. Web Service Modeling Ontology (WSMO)

The Web service Modeling Ontology (WSMO) [28] [65] is an ontology based Web service composition approach used to solve integration problems dynamically. Its conceptual design is made from Web Service Modeling Framework WSMF [48]. WSMF has four elements: Ontologies, Web services, goals, and mediators. The WSMO introduces the addition of non-functional characteristics which can be used by all four modeling elements. WSMO in collaboration with Web Service Modeling Language

WSML [29] allows to write the annotations of Web services according to conceptual model, and using Web Service Execution Environment WSMX [38] with WSMO it allows the dynamic discovery, selection, mediation, and invocation of Web services. WSMO recommends the use of vocabularies. WSMO supports the definition of multiple interfaces for single Web service.

2.7 Chapter Summary

This chapter started with introduction of SOA and then reviewed Web services and its components. It reviewed the literature of highly relevant SLA approaches, SLA elements, SLA lifecycle stages, Cloud computing service providers and QoS Terms. The SLA elements and SLA lifecycle stages were collected together from different approaches, rearranged and then finally refined in order to be more structured and useful for Cloud computing services.

This chapter included the most popular Cloud service providers and their SLAs, and analyzed the use of SLA terms by their services. The SLA terms used by Cloud service providers were classified into implicit and explicit usage followed by some recommendations of SLA elements to be included by this thesis.

This chapter has reviewed the QoS terms from TRS and SLAs and its relevant approaches were analyzed and further classified into two categories: QoS Terms defined without SLAs and QoS Terms defined within the SLAs. It has reviewed different QoS selection techniques, collected the different drawbacks and limitations of different approaches against the Fuzzy Logic. The advantages of Fuzzy logic and its use by different approaches are also reviewed in this chapter.

The chapter also reviewed and determined that the inherent model of Web services and specifically the computational Cloud service providers lack standardized QoS management and monitoring features using SLAs during Web service composition. Finally, the Web service composition approaches were discussed for problems requiring a composite set of services.

3. Concept Elements

3.1 Introduction

For the structure, management and monitoring of Service Level Agreements (SLAs) during the composition of services in order to fulfill the composite consumer requirements with good Quality of Service (QoS), the individual in depth concept elements are defined in this chapter. The concept elements are defined for services, parties, SLAs and QoS terms. The ontological representation of SLA elements and QoS terms is also given in this chapter. The concept elements will be used in the proposed framework of this thesis in order to answer the research question and solve the problems addressed in this thesis.

3.2 Basic Concept Definitions

The generic components are defined in the form of basic concept definitions for SLA elements and QoS terms. The basic concept definitions for SLA elements are defined by taking elements from Table 2-6 which are shown in Table 3-1 to Table 3-7. The definition of basic concept element for QoS terms is defined by taking QoS terms from Table 2-33 (QoS Terms defined without the SLA Parameters) and Table 2-42 (QoS Terms defined within SLA Parameters) which is shown in Table 3-8. Some additional concept definitions are also added by this thesis for proposed framework usage (i.e. representing consumer requirements etc) that are shown in Table 3-9.

Table 3-1: Basic Concept Elements for SLA Elements: Name

Element No.	Element Name	Tuple/Set	Short Name	Has Sub Element
1	Agreement Name	Set	AN	No
2	Agreement Template	Set	ATEMP	No
3	Agreement Terms	Set	ATER	No

Table 3-2: Basic Concept Elements for SLA Elements: Purpose

Element No.	Element Name	Tuple/Set	Short Name	Has Sub Element
1	Purpose	Set	SLAPU	No

Table 3-3: Basic Concept Elements for SLA Elements: Validity Period

Element No.	Element Name	Tuple/Set	Short Name	Has Sub Element
1	Validity Period	Tuple	SLAVP	No

Table 3-4: Basic Concept Elements for SLA Elements: Parties

Element No.	Element Name	Tuple/Set	Short Name	Has Sub Element
1	Service Provider	Set	SP	No
2	Service Consumer	Set	SC	No
3	Third Party	Set	TP	No

Table 3-5: Basic Concept Elements for SLA Elements: Party Roles

Element No.	Element Name	Tuple/Set	Short Name	Has Sub Element
1	Signatory Party	Set	SIP	No
2	Supporting Party	Set	SUP	No
3	Agreement Initiator	Set	AIN	No
4	Agreement Responder	Set	ARES	No

Table 3-6: Basic Concept Elements for SLA Elements: Service Terms

Element No.	Element Name	Tuple/Set	Short Name	Has Sub Element
1	Service Description Terms	Tuple	SDT	No
2	Service Properties	Set	SPRO	No
3	SLA Parameter	Set	SLAPAR	No
4	Metric	Set	MET	No
5	Measurement Directive	Set	MD	No
6	Function	Set	FUN	No
7	Any Attribute	Set	AA	Yes

Table 3-7: Basic Concept Elements for SLA Elements: Guarantee Terms

Element No.	Element Name	Tuple/Set	Short Name	Has Sub Element
1	Obligations	Set	OB	Yes
2	Service Scope	Set	SCOP	Yes
3	Service Level Objective	Set	SLO	Yes
4	Action Guarantee	Set	AG	Yes
5	Penalties	Set	PEN	Yes
6	Optional Services	Set	OS	Yes
7	Restrictions	Set	RES	Yes
8	Exclusions	Set	EXC	Yes

Table 3-8: Basic Concept Element for QoS Term metrics

Element No.	Element Name	Tuple/Set	Short Name	Has Sub Element
1	QoS Term	Set	QoS_Term	No

Table 3-9: Basic Concept Elements added by Thesis for SLAAgent Framework

Element No.	Element Name	Tuple/Set	Short Name	Has Sub Element
1	Requirement	Set	REQ	Yes
2	Service	Set	S	No
3	Inputs to Service	Set	I	No
4	Outputs of Service	Set	O	No
5	Service Operations	Set	SOP	No
6	Composite Service	Set	CS	No
7	SLA Start Date	Tuple	SLASD	No
8	SLA End Date	Tuple	SLAED	No
9	QoS Score	Set	QoS_Score	No

Definition 1 (Agreement Name): The name of the *SLA Agreement*, denoted by *AN* is a set described by a single element of string value is denoted by:

$$AN = \{\text{"Agreement Name"}\}$$

Definition 2 (Agreement Template): The information about the type of an Agreement, denoted by *ATEMP* is a set of *atempn* individual $ATEMP_i$ Agreement Template attributes, and $1 \leq i \leq atempn$.

$$ATEMP = \{ATEMP_1, ATEMP_2, \ldots ATEMP_{atempn}\}$$

Definition 3 (Agreement Terms): The term definitions of an Agreement denoted by *ATER* is a set of *atern* individual $ATER_i$ Agreement Terms, and $1 \leq i \leq atern$.

$$ATER = \{ATER_1, ATER_2, \ldots ATER_{atern}\}$$

Definition 4 (Purpose): The description or purpose of creating the *SLA*, denoted by *SLAPU* is a set described by a single element of string value is denoted by:

$$SLAPU = \{\text{"Purpose of SLA"}\}$$

Definition 5 (Validity Period): The time duration of an SLA denoted by *SLAVP* is a tuple of SLA Validity Period comprising of the tuples of SLA Start Date denoted by *SLASD* and SLA End Date denoted by *SLAED.*

$$SLAVP= (SLASD, SLAED)$$

Definition 6 (SLA Start Date): The particular *Starting Date* of an *SLA* denoted by *SLASD* is a tuple of SLA Start Date comprising of the Time denoted by *T* in 24 Hour format, the Day denoted by *D*, the month denoted by *M* and the year denoted by *Y*.

$$SLASD= (T, D, M, Y)$$

Definition 7 (SLA End Date): The particular Ending Date of an SLA denoted by *SLAED* is a tuple of SLA End Date comprising of the Time denoted by *T* in 24 Hour format, the Day denoted by *D*, the month denoted by *M* and the year denoted by *Y*.

$$SLAED= (T, D, M, Y)$$

Definition 8 (Parties): The Parties of an SLA denoted by *PAR* is a tuple of SLA Parties comprising of the sets of Service Provider denoted by *SP,* Service Consumer denoted by *SC* and Third Party denoted by *TP*.

$$PAR= (SP, SC, TP)$$

Definition 9 (Service Provider): The Provider of the Service or Services (also called Service Provider) denoted by *SP* is a set of *spn* individual SP_i Service Providers, and $1 \leq i \leq spn$.

$$SP= \{SP_1, SP_2, \ldots SP_{spn}\}$$

Definition 10 (Service Consumer): The consumer of the Service or Services (also called Service Consumer) denoted by *SC* is a set of *scn* individual SC_i Service Consumers, and $1 \leq i \leq scn$.

$$SC= \{SC_1, SC_2, \ldots, SC_{scn}\}$$

Definition 11 (Third Party): The Third Party involved in an SLA denoted by *TP* is a set of *tpn* individual TP_i Third Parties, and $1 \leq i \leq tpn$.

$$TP= \{TP_1, TP_2, \ldots TP_{tpn}\}$$

Definition 12 (Party Role): The Party Role of an SLA denoted by *PR* is a tuple of SLA Party Roles comprising of the sets of Signatory Party denoted by *SIP*, Supporting Party denoted by *SUP*, Agreement Initiator denoted by *AIN* and Agreement Responder *ARES.*

$$PR= (SIP,SUP,AIN,ARES)$$

Definition 13 (Signatory Party): The Signatory Party involved in an SLA denoted by *SIP* is a set of *sipn* individual SIP_i Signatory Parties, and $1 \leq i \leq sipn.$

$$SIP= \{SIP_1,SIP_{2,} \ldots SIP_{sipn} \}$$

Definition 14 (Supporting Party): The supporting Party involved in an SLA denoted by *SUP* is a set of *supn* individual SUP_i Supporting Parties and $1 \leq i \leq supn.$

$$SUP= \{SUP_1,SUP_{2,} \ldots SUP_{supn} \}$$

Definition 15 (Agreement Initiator): The Agreement Initiator involved in an SLA denoted by *AIN* is a set of *ainn* individual AIN_i Agreement Initiators, and $1 \leq i \leq ainn.$

$$AIN= \{AIN_1, AIN_{2,} \ldots AIN_{ainn}\}$$

Definition 16 (Agreement Responder): The Agreement Responder involved in an SLA denoted by *ARES* is a set of *aresn* individual $ARES_i$ Agreement Responders, and $1 \leq i \leq aresn.$

$$ARES= \{ ARES_1, ARES_{2,} \ldots ARES_{aresn} \}$$

Definition 17 (Service Terms): The Service Terms of an SLA denoted by *ST* is a tuple of SLA Service Terms comprising of a tuple of Service Description Terms denoted by *SDT* and sets of Service Properties denoted by *SPRO,* SLA Parameter denoted by *SLAPAR*, Metric denoted by *MET*, Measurement Directive denoted by *MD*, Function denoted by *FUN* and Any Attributed denoted by *AA.*

$$ST= (SDT, SPRO, SLAPAR, MET, MD, FUN, AA)$$

Definition 18 (Service Description Terms): The elements of Service denoted by *SDT* is a tuple of Service Description Terms comprising of the set of Services denoted by S,

set of Inputs to Service denoted by *I*, the set of Service Operations denoted by *SOP* and the set of Outputs of Service denoted by *O*.

$$SDT= (S, I, SOP, O)$$

Definition 19 (Service): The Service or number of Services involved in an SLA denoted by *S* is a set of *sn* individual S_i Services, and $1 \leq i \leq sn$.

$$S= \{S_1, S_2, \ldots S_{sn} \}$$

Definition 20 (Inputs to Service): The input parameters passed to the Service denoted by *I* is a set of *in* individual I_{in} Inputs to Services, and $1 \leq i \leq in$.

$$I = \{I_1, I_2,..., I_{in} \}$$

Definition 21 (Outputs of Service): The output parameters of Service denoted by *O* is a set of *on* individual O_{on} Outputs of Services, and $1 \leq i \leq on$.

$$O = \{O_1, O_2,..., O_{on} \}$$

Definition 22 (Service Operations): The operations performed by Service denoted by *SOP* is a set of *sopn* individual SOP_i Services Operations, and $1 \leq i \leq sopn$.

$$SOP = \{SOP_1, SOP_2,.....,SOP_{sopn}\}$$

Definition 23 (Composite Service): The group of Services also called Composite Services involved in an SLA denoted by *CS* is a set of *csn* individual CS_i Composite Services, and $1 \leq i \leq csn$.

$$CS= \{CS_1, CS_2,..., CS_{csn}\}$$

Definition 24 (Service Properties): The Service properties that explain domain specific features of a Service denoted by *SPRO* is a set of *spron* individual $SPRO_i$ Service Properties, and $1 \leq i \leq spron$.

$$SPRO= \{SPRO_1, SPRO_2 , \ldots , SPRO_{spron}\}$$

Definition 25 (SLAParameter): The properties of a Service Object pointing to a composite Metric denoted by *SLAPAR* is a set of *slaparn* individual $SLAPAR_i$ SLA Parameters, and $1 \leq i \leq slaparn$.

$$SLAPAR = \{SLAPAR_1, SLAPAR_2 , \ldots , SLAPAR_{slaparn}\}$$

Definition 26 (Metric): The details about the values of Service Properties denoted by *MET* is a set of *metn* individual MET_i Metrics, and $1 \leq i \leq metn$.

$$MET = \{ MET_1, MET_2 , \ldots , MET_{metn}\}$$

Definition 27 (Measurement Directive): The explanation about how the parameter values should be measured denoted by *MD* is a set of *mdn* individual MD_i Measurement Directives, and $1 \leq i \leq mdn$.

$$MD = \{MD_1, MD_2 , \ldots , MD_{mdn}\}$$

Definition 28 (Function): The measurement algorithm for computing composite metrics denoted by *FUN* is a set of *funn* individual FUN_i Functions, and $1 \leq i \leq funn$.

$$FUN = \{FUN_1, FUN_2 , \ldots , FUN_{funn}\}$$

Definition 29 (Any Attribute): The additional attribute specified in an agreement denoted by *AA* is a set of *aan* individual AA_i Any Attribute, and $1 \leq i \leq aan$.

$$AA = \{AA_1, AA_2 , \ldots , AA_{aan}\}$$

The sub parts of additional attributes denoted by *Sub_AA* is a set of *sub_aan* individual Sub_AA_i sub groups of Any Attributes, and $1 \leq i \leq sub\_aan$.

$$Sub\_AA = \{Sub\_AA_1, Sub\_AA_2, \ldots , Sub\_AA_{sub\_aan}\}$$

Definition 30 (Guarantee Terms): The Guarantee Terms of an SLA denoted by *GT* is a tuple of Obligations denoted by *OB*, Service Scope denoted by *SCOP*, Service Level Objective denoted by *SLO*, Action Guarantee denoted by *AG*, Penalties denoted by *PEN*, Optional Services denoted by *OS*, Restrictions denoted by *RES* and Exclusions denoted by *EXC*.

$$GT = (OB, SCOP, SLO, AG, PEN, OS, RES, EXC)$$

Definition 31 (Obligations): The guarantees and constraints imposed on SLA parameters denoted by *OB* is a set of *obn* individual OB_i Obligations, and $1 \leq i \leq obn$.

$$OB = \{OB_1, OB_2 , \ldots , OB_{obn}\}$$

Definition 32 (Service Scope): The range of Service boundary covered by Service Provider denoted by *SCOP* is a set of *scopn* individual $SCOP_i$ Service Scopes, and $1 \leq i \leq scopn$.

$$SCOP = \{SCOP_1, SCOP_2, \ldots, SCOP_{scopn}\}$$

The sub parts of range of Service boundaries covered by Service Providers denoted by *Sub_SCOP* is a set of *sub_scopn* individual Sub_SCOP_i sub groups of Service Scopes, and $1 \leq i \leq sub\_scopn$.

$$Sub\_SCOP = \{Sub\_SCOP_1, Sub\_SCOP_2, \ldots, Sub\_SCOP_{sub\_scopn}\}$$

Definition 33 (Service Level Objectives): The objectives that must be fulfilled from Services denoted by *SLO* is a set of *slon* individual SLO_i Service Level Objectives, and $1 \leq i \leq slon$.

$$SLO = \{SLO_1, SLO_2, \ldots, SLO_{slo}\}$$

The sub parts of an objective that must be fulfilled by Services denoted by *Sub_SLO* is a set of *sub_slon* individual Sub_SLO_i sub groups of Service Level Objectives, and $1 \leq i \leq sub\_slon$.

$$Sub\_SLO = \{Sub\_SLO_1, Sub\_SLO_2, \ldots, Sub\_SLO_{sub\_slon}\}$$

Definition 34 (Action Guarantee): The actions that must be guaranteed performed by Service Providers denoted by *AG* is a set of *agn* individual AG_i Action Guarantees, and $1 \leq i \leq agn$.

$$AG = \{AG_1, AG_2, \ldots, AG_{ago}\}$$

The sub parts of an action guarantee that must be performed by Service Providers denoted by *Sub_AG* is a set of *sub_agn* individual Sub_AG_i sub groups of Action Guarantees, and $1 \leq i \leq sub\_agn$.

$$Sub\_AG = \{Sub\_AG_1, Sub\_AG_2, \ldots, Sub\_AG_{sub\_gn}\}$$

Definition 35 (Penalties): The incomplete Service provided will result in payment of fine or compensation denoted by *PEN* is a set of *penn* individual PEN_i SLA Penalties, and $1 \leq i \leq penn$.

$$PEN= \{PEN_1, PEN_2 , . . . , PEN_{penn}\}$$

The sub parts of an incomplete Service provision that will result in payment of fine or compensation denoted by *Sub_PEN* is a set of *sub_penn* individual Sub_PEN_i sub groups of SLA Penalties, and $1 \leq i \leq sub\_penn$.

$$Sub\_PEN= \{Sub\_PEN_1, Sub\_PEN_2 , . . . , Sub\_PEN_{sub\_penn}\}$$

Definition 36 (Optional Service): The services that are not part of an SLA, but provided on demand denoted by *OS* is a set of *osn* individual OS_i Optional Services, and $1 \leq i \leq osn$.

$$OS= \{OS_1, OS_2 , . . . , OS_{osn}\}$$

The sub parts of services that are not part of an SLA but provided on demand denoted by *Sub_OS* is a set of *sub_osn* individual Sub_OS_i sub groups of Optional Services, and $1 \leq i \leq sub\_osn$.

$$Sub\_OS= \{Sub\_OS_1, Sub\_OS_2 , . . . , Sub\_OS_{sub\_os}\}$$

Definition 37 (Restrictions): The necessary prohibition of actions defined in an SLA about the Service denoted by *RES* is a set of *resn* individual RES_i Service Restrictions, and $1 \leq I \leq resn$.

$$RES= \{ RES_1, RES_2 , . . . , RES_{resn}\}$$

The sub parts of necessary prohibitions of actions defined in an SLA about the Services denoted by *Sub_RES* is a set of *sub_resn* individual Sub_RES_i sub groups of Service Restrictions, and $1 \leq i \leq sub\_resn$.

$$Sub\_RES= \{Sub\_RES_1, Sub\_RES_2 , . . . , Sub\_RES_{sub\_resn}\}$$

Definition 38 (Exclusions): The explicitly specified non-covering items in an SLA denoted by *EXC* is a set of *excn* individual EXC_i Service Exclusions, and $1 \leq i \leq excn$.

$$EXC = \{ EXC_1, EXC_2 , . . . , EXC_{excn}\}$$

The sub parts of explicitly specified non-covering items in an SLA denoted by *Sub_EXC* is a set of *sub_excn* individual Sub_EXC_i sub groups of Service Exclusions, and $1 \leq i \leq sub\_excn$.

$$Sub\_EXC = \{Sub\_EXC_1, Sub\_EXC_2 , \ldots , Sub\_EXC_{sub\_excn}\}$$

Definition 39 (QoS Terms): The QoS Terms denoted by *QoS_TERM* is a set of *qos_termn* individual QoS_TERM_i Quality of Service Terms (defined within and without SLAs), and $1 \leq i \leq qos\_termn$.

$$QoS\_TERM = \{ QoS\_TERM_1, QoS\_TERM_2 , \ldots , QoS\_TERM_{qos\_termn}\}$$

Definition 40 (QoS Score): The QoS Score denoted by *QoS_Score* is a set of *qos_scoren* individual QoS_SCORE_i Quality of Service Score (defined within and without SLAs), and $1 \leq i \leq qos\_scoren$.

$$QoS\_SCORE = \{ QoS\_SCORE_1, QoS\_SCORE_2 , \ldots , QoS\_SCORE_{qos\_scoren}\}$$

Definition 41 (Requirement): The Requirement of Service Consumers or the Services required to Service Consumers denoted by *REQ* is a set of *reqn* individual REQ_i Required Services denoted by S_i, where $1 \leq i \leq reqn$.

$$REQ = \{REQ_1, REQ_2, \ldots, REQ_{reqn}\}$$

The sub parts of Service Consumer Requirements denoted by *Sub_REQ* is a set of *sub_reqn* individual Sub_REQ_i sub groups of Consumer Requirements, and $1 \leq i \leq sub\_reqn$.

$$Sub\_REQ = \{Sub\_REQ_1, Sub\_REQ_2, \ldots , Sub\_REQ_{sub\_reqn}\}$$

3.3 SLA Ontology

The refined SLA elements from Table 2-6 are represented as ontologies using Protégé [74] in Figure 3-1 to Figure 3-8.

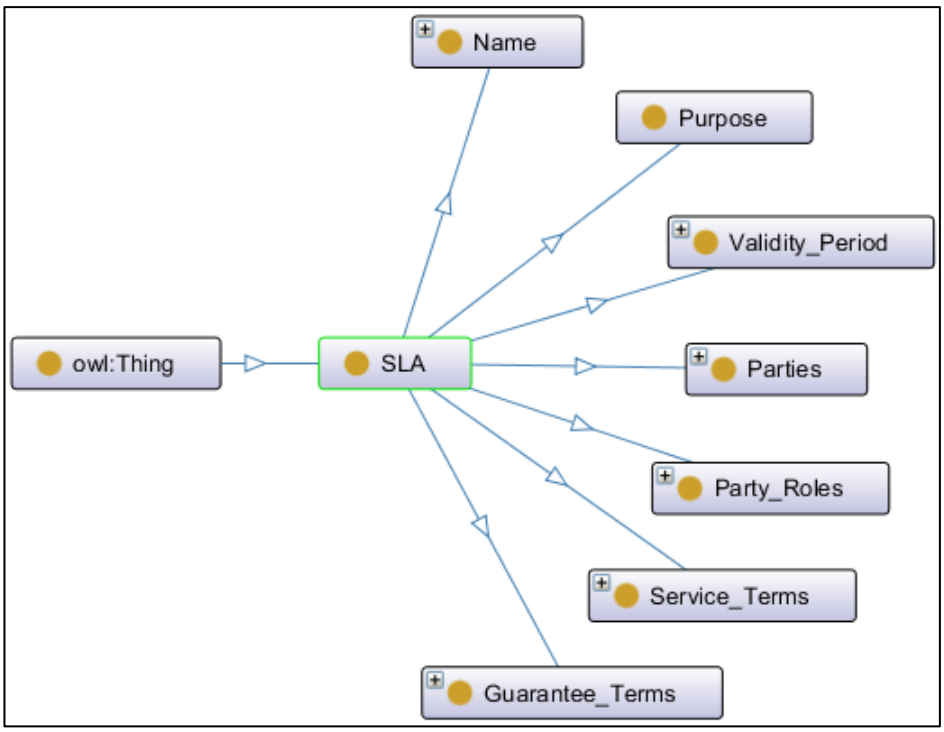

Figure 3-1: SLA Ontology-Main Elements

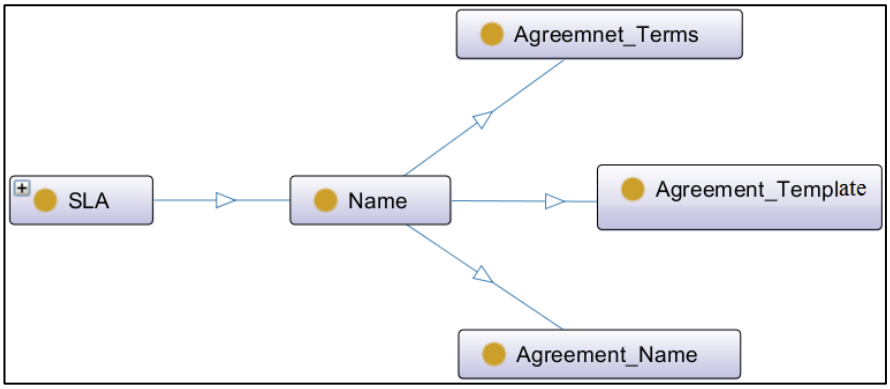

Figure 3-2: SLA Ontology-Sub Element: Name

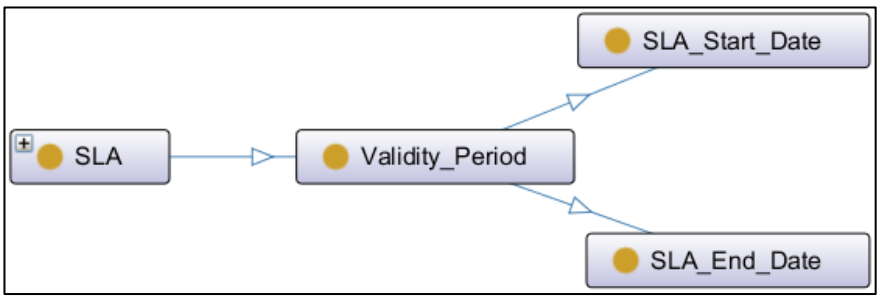

Figure 3-3: SLA Ontology-Sub Element: Validity Period

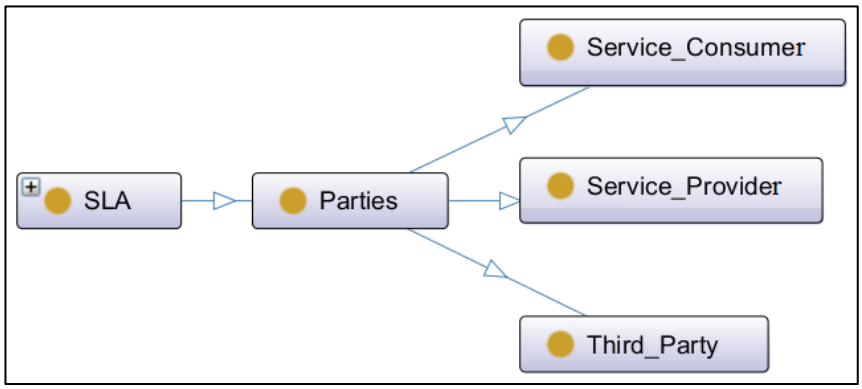

Figure 3-4: SLA Ontology-Sub Element: Parties

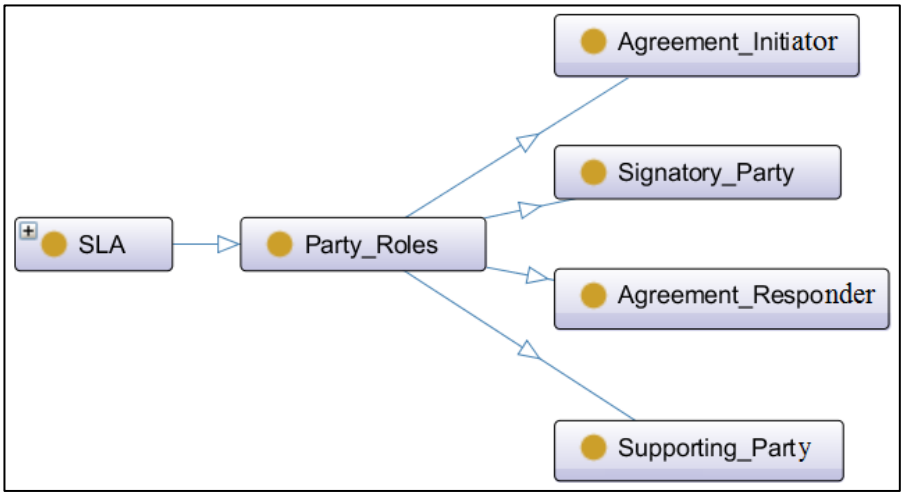

Figure 3-5: SLA Ontology-Sub Element: Party Roles

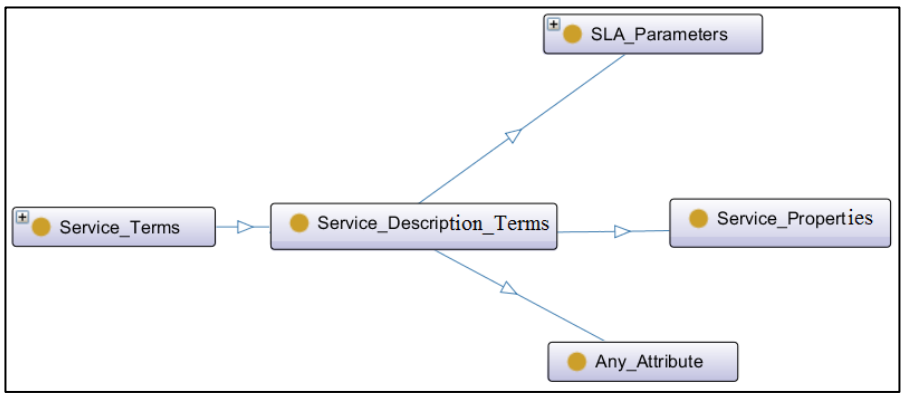

Figure 3-6: SLA Ontology-Sub Element: Service Terms

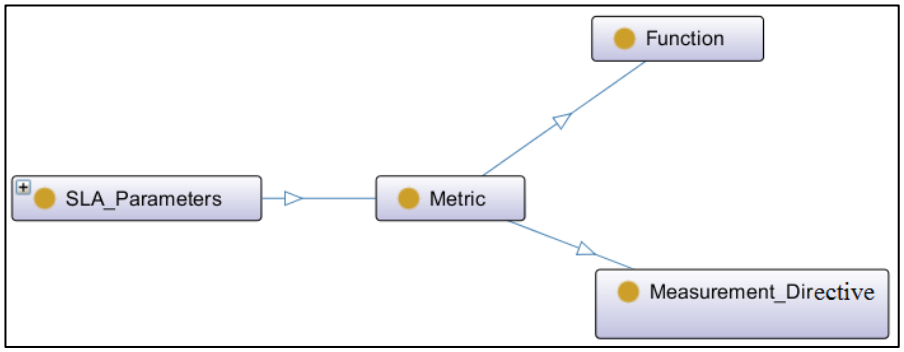

Figure 3-7: SLA Ontology-Sub Element: SLA Parameters

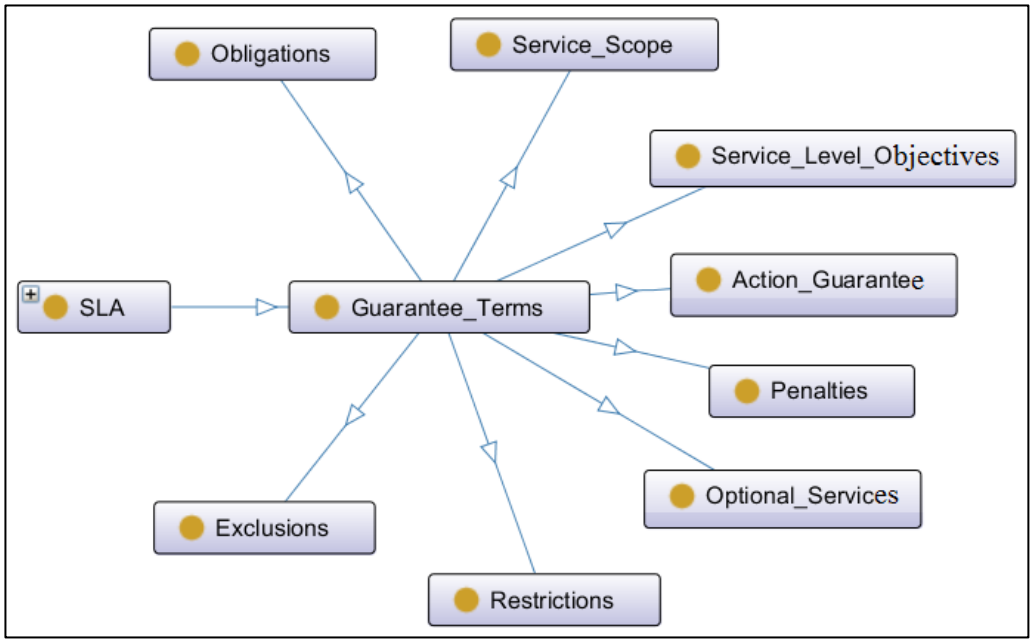

Figure 3-8: SLA Ontology-Sub Element: Guarantee Terms

3.4 QoS Ontology

The ontological representation of QoS Terms from Table 2-43 are given below in Figure 3-9 to Figure 3-15.

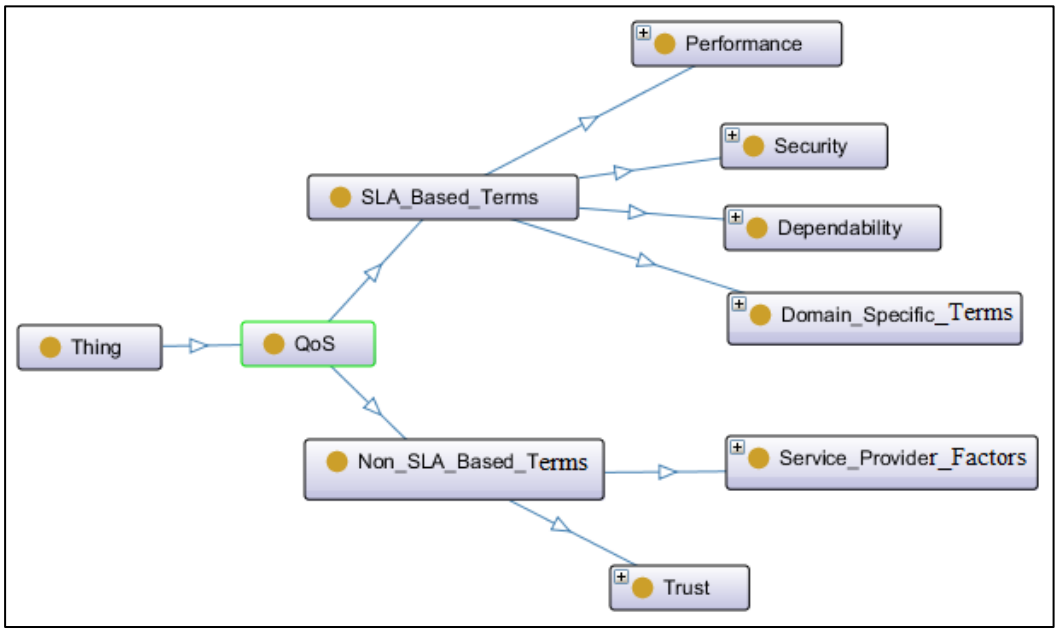

Figure 3-9: QoS Main Ontology

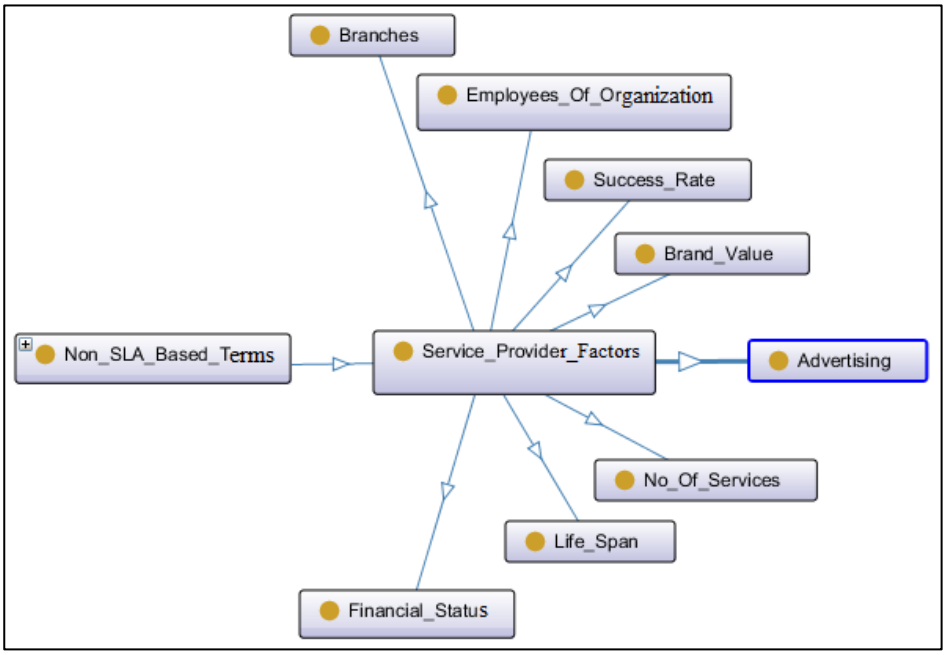

Figure 3-10: QoS Sub-Ontology: Service Provider Factors

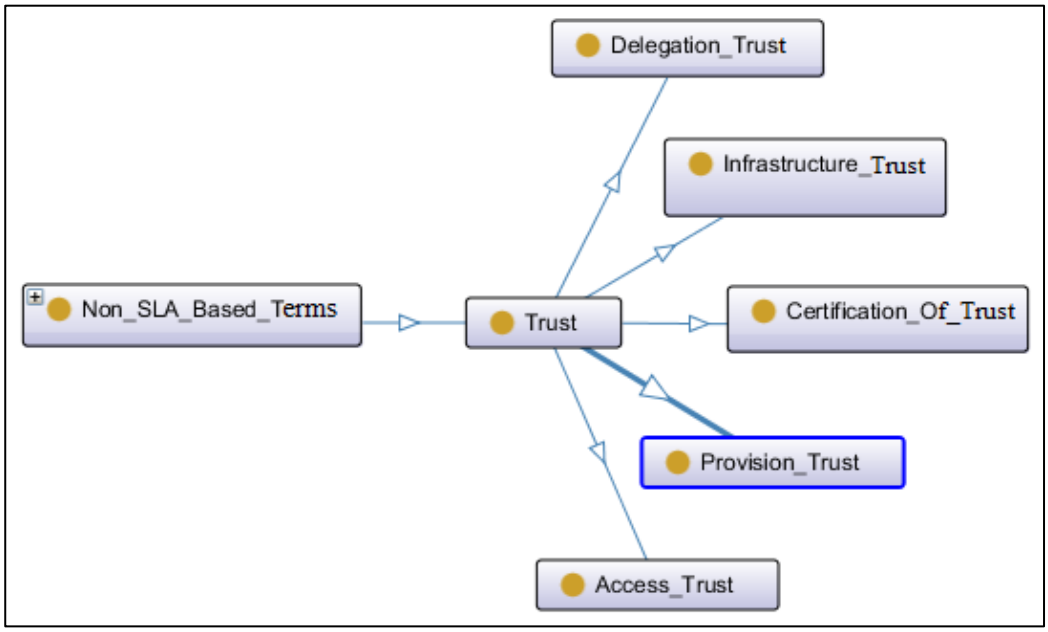

Figure 3-11: QoS Sub-Ontology: Trust

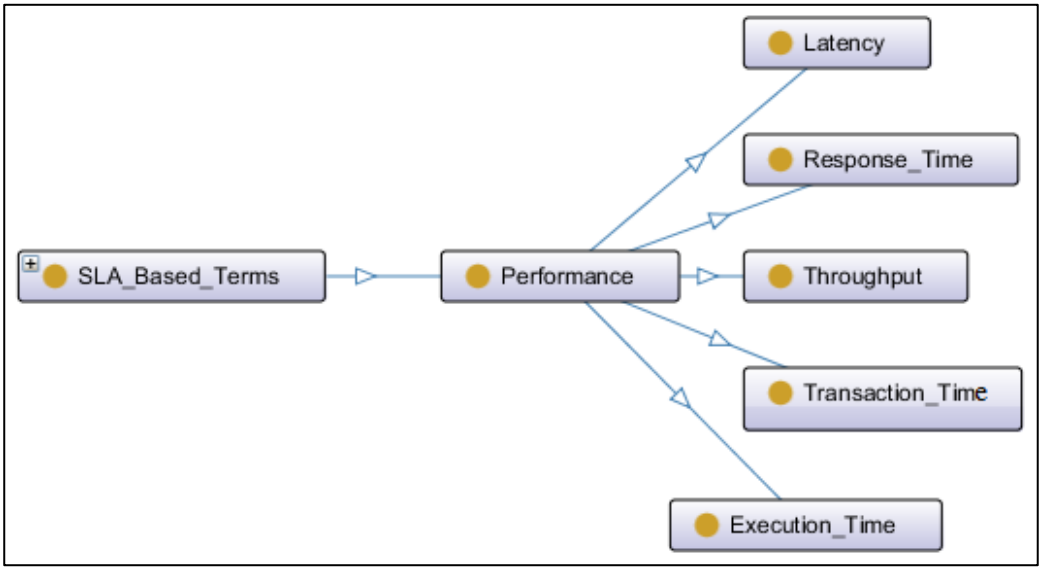

Figure 3-12: QoS Sub-Ontology: Performance

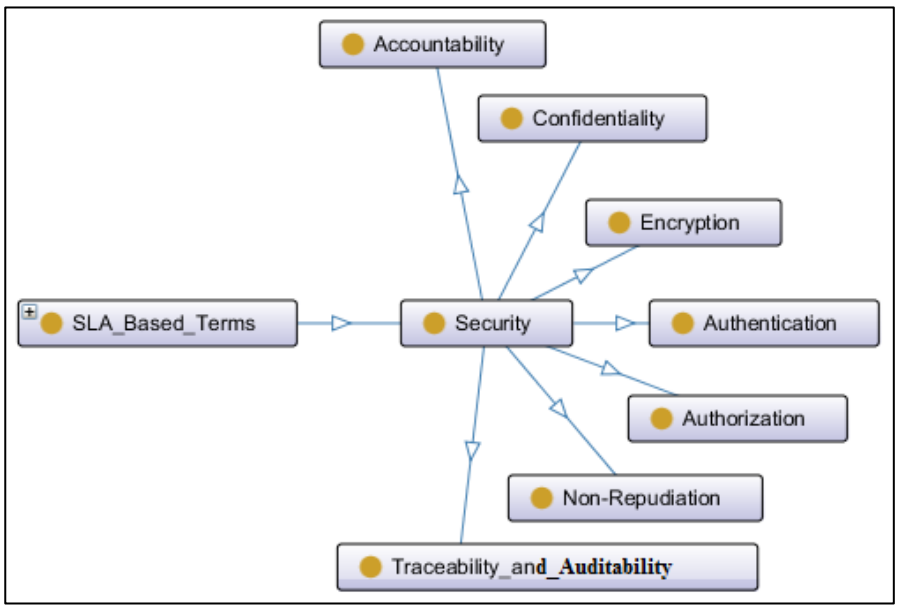

Figure 3-13: QoS-Sub Ontology: Security

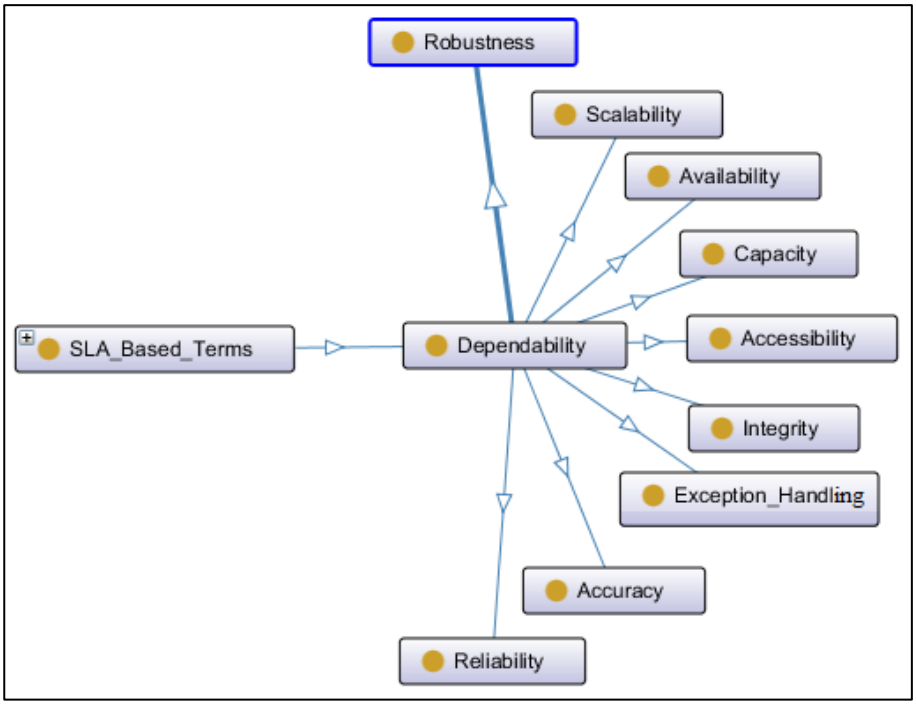

Figure 3-14: QoS-Sub Ontology: Dependability

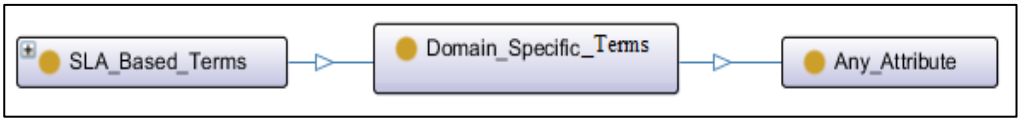

Figure 3-15: QoS-Sub Ontology: Domain Specific Terms

3.5 Chapter Summary

This chapter defined the important basic concept elements in the form of Sets and Tuples used for the structure, management and monitoring of SLAs. This chapter classified the basic concept elements into different sections for SLA elements, QoS terms and some additional basic concept elements for usage in proposed framework of this thesis. This chapter also contained the Ontological representation of SLA elements and QoS terms that are used in next chapters for framework understanding and usage.

4. Quality of Service Calculation

4.1 Introduction

The selection of a suitable service required by a consumer is a challenge when various similar or equivalent services are available in the same domain of interest. Services with similar functionalities can have different Quality of Service (QoS). The Quality of Service can help with ranking services among the group of similar functional services that are different in qualitative characteristics. Quality of Service describes how well a service performs its operations or satisfies the consumer requirements. The non-functional aspects of a service are also known as the Quality of Service. This chapter discusses the generic formulas for calculating the Quality of Service as reputation of service provider. The services selection criterion is based on the ranking of the services accumulated from the QoS obtained with and without the help of SLAs, considering some technical and domain specific terms. Selection of Quality of Service attributes can vary from time to time and depend on the selection by consumers as well as providers. Quality of Service should be monitored frequently with regular intervals so that the ranking of the services remain updated.

Due to dynamic nature of service performance and reputation of service providers, Fuzzy Inference Systems are best suited to calculate the QoS from unclear or less precise data of QoS values by the monitoring the service performance and reputation of services providers. This chapter explores the QoS accumulation with the help of Fuzzy Inference Systems based QoS Term metrics defined within and without SLA Parameters.

4.2 Fuzzy Sets and Fuzzy Logic

In classical set theory (a crisp or an ordinary set) the membership of elements is defined in binary terms, which states that an element either belongs to the set or does not belong to the set. In Fuzzy set theory, the membership of an element is described using a membership function within the values in the range of real unit interval [0, 1]. Fuzzy sets were proposed by Zadeh [130] in 1965. The Fuzzy set theory is used in the domain areas where the information is imprecise or incomplete.

A membership function representing a Fuzzy set A on the universe of discourse X can be denoted as μ_A: X → [0, 1]. Each element of X (in the input space) is mapped to a membership value between (0 and 1) which is also known as degree or grade of membership of the elements in X that quantifies to the Fuzzy set A.

The Fuzzy sets are represented graphically with the help of Membership Functions. The universe of discourse (input space) is represented on the x-axis and the degree of membership is represented on the y-axis between the interval value of [0, 1]. The most common used membership functions used for Fuzzy Sets are Trapezoidal, Triangular and Gaussian Functions [131].

A Fuzzy set is represented with the degree of the membership of an element while the particular range of degree is represented with Fuzzy linguistic variable in a Fuzzy set. The Fuzzy linguistic variables have a meaning and are labelled with a word or sentence in a language. Figure 4-1 shows the membership functions of Ages represented with Fuzzy Sets “young age”, “middle age” and “old age”, it shows the age of a person on horizontal axis of the graphs the “age” in year from the universal set and the degree of the age to which a person can belong is represented on the vertical axis divided into “young age”, “middle age” and “old age”. Thus the graph representation is the Fuzzy set membership of groups of people falling into the age groups of young age, middle age and old age. It is also important to note from Figure 4-1, that there is overlapping of membership function values, this indicates that using Fuzzy Logic theory, certain decision making is required to determine the approximate match of crisp value to the Fuzzy set membership.

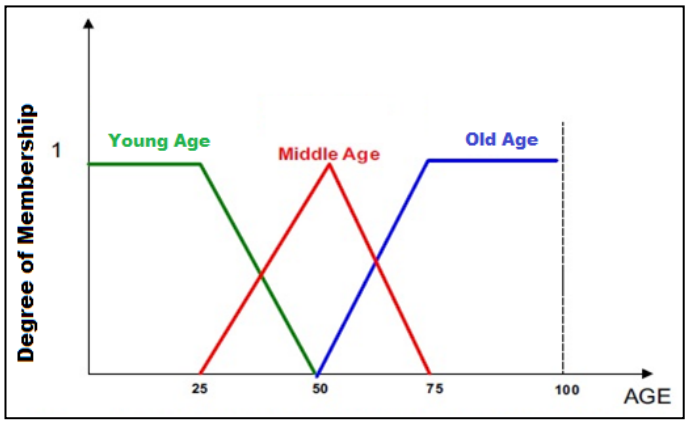

Figure 4-1: Membership Functions of Fuzzy Sets Young, Middle and Old Age

Fuzzy logic deal with reasoning in the approximate form rather than exact or fixed values using a many-valued logic structure. A many-valued logic is a propositional calculus in which more than two truth values are used (it may range between 0 and 1), while in binary sets variables may only take false or true values [91]. Fuzzy Logic is a control system methodology for problem solving. It gives a simple method to draw a definite result based upon ambiguous, imprecise, vague, noisy or missing input information.

4.3 Fuzzy Inference System

A Fuzzy Inference System (FIS) is used to map an input space to an output space with the help of the theory of Fuzzy sets in order to solve decision-making problems. A FIS aims to formalize the reasoning process used in human language. In order to reason about data, the FIS uses collection of Fuzzy membership functions and Fuzzy rules instead of Boolean logic. The rules used in the FIS are in the form of "If-then" Fuzzy production rule statements. The antecedent of the rule specifies to what degree the rule is applicable, while the conclusion allocates the Fuzzy function to each of the one or more output Fuzzy variables. In a FIS there may exist more than one conclusion per rule. The collection of rules in FIS is known as the Knowledge base.

Fuzzy Inference System models are classified according to the membership function approximation. They are divided into *Non-Additive* and *Additive* rule models. The *Non-Additive* model has Mamdani Model [83] while the *Additive Rule* models have Takagi-Sugeno-Kan (TSK) model [113], Standard Additive (Kosko) model [75] and Tsukamoto model [121]. The most popular methods in Fuzzy Inference used are the Mamdani and Sugeno methods. The first two parts in both methods i.e. fuzzifying the inputs and combining the antecedent part of the Fuzzy rule using Fuzzy operators are same, while the rest of the steps are different.

4.3.1 Components of Fuzzy Inference System

A Fuzzy Inference System is composed of four general components:

Fuzzifier: The Fuzzifier transforms the system inputs also known as crisp inputs into Fuzzy sets, using Fuzzy membership functions.

Fuzzy Knowledge base: The Fuzzy knowledge base stores the Fuzzy rules in the form of If-then structures.

Inference Engine: the inference engine simulates the process of human reasoning using Fuzzy Inference on the basis of If-Then rules and given inputs. The "If-Then" rule has two parts: antecedent and consequent.

Defuzzifier: The Defuzzifier transforms the Fuzzy set given by the outcome of Inference engine into a crisp value.

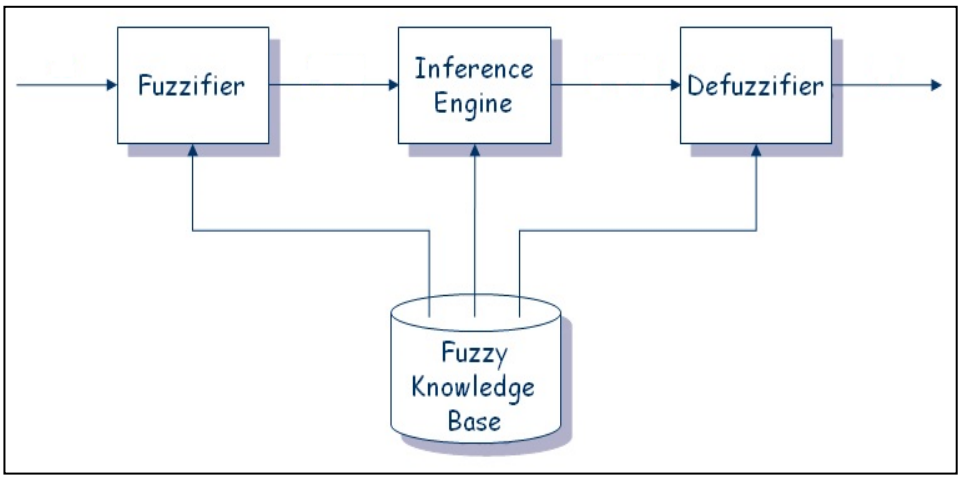

Figure 4-2: Fuzzy Inference System

4.3.2 Fuzzy Inference Process

The Fuzzy Inference Process involves the following main functional steps:

1. Determine the Number of Inputs and Outputs
2. Define Input Membership Functions
3. Define Output Membership Functions
4. Determine the Number of Fuzzy rules
5. Fuzzify inputs
6. Combining the Fuzzified inputs
7. Compute the rule strength
8. Define Fuzzy Associative Memory

9. Find Consequence of the Rule
10. Aggregate Rule Outputs
11. Defuzzify output

Step 1: Determine the Number of Inputs and Outputs

In this step, determine the number inputs to the Fuzzy Inference System and expected number of outputs. Usually there is one output and two or more inputs used for Fuzzy Inference Process.

Step 2: Define Input Membership Functions

In this step, depending on the number of inputs define Input Membership Functions for each input crisp variable. Each Input Membership function can have a separate range of crisp set input crisp values, while the output is the degree of membership from a Fuzzy set values between (0 and 1) along with Fuzzy linguistic variable (e.g. Low, Medium and High).

Step 3: Define Output Membership Functions

In this step define Output Membership Functions depending on requirement of the Fuzzy Inference System. Usually there is one Output Membership function used based on the different number of crisp inputs. Each Output Membership function can have a separate range of values. An Output Membership function can have a range of input values from the crisp set or Fuzzy set, while the output is the degree of membership from a Fuzzy set values between (0 and 1) along with Fuzzy linguistic variable (e.g. Low, Medium and High).

Step 4: Determining a set of Fuzzy Rules

In the construction of a Fuzzy Inference System, Fuzzy rules are the linguistic statements that describe how the Fuzzy Inference System should produce the output with regards to the number of inputs. The total number of Fuzzy rules depends on the number of input variables and the number of Fuzzy linguistic variables in each input membership function. But only those rules will be used for the Fuzzy Inference Process

which will fall within the range of degree of membership with possible overlapping or no overlapping of the degree of membership function values in any rule.

Step 5: Fuzzify Inputs

In this step the inputs are passed to the membership function in order to determine the degree to which they belong to each of the appropriate Fuzzy sets. The inputs are always in the form of crisp numerical value restricted to the universe of discourse of the input variables usually within the interval between 0 and 10, while the output of the membership function is a Fuzzy degree of membership between the interval of 0 and 1. Fuzzification on inputs usually results from a membership function or using a lookup table.

Step 6: Combine Fuzzified Inputs

After fuzzifying the inputs, the degree of antecedent becomes known to which Fuzzy set it is satisfied in each rule. But the antecedent of a rule can have more than one parts, therefore the antecedent of the rule is converted into single membership value by using the Fuzzy operators such as: AND operator (also known as *t-norm* for intersection*)*, and OR operator (also called *t-conorm* for union). These Fuzzy operators work on two or more inputs from the antecedent part for the fuzzified input variables, but these Fuzzy operators give only a single value as output. There can be different ways to compute the *AND* or *OR* [7]. The *min* (minimum) and *prod* (product) are two methods supported by *AND* operator, while *max* (maximum) and *probor* (the probabilistic OR) is supported by *OR* operator. The *probor* method is implemented using algebraic sum [6].

Step 7: Compute Rule Strength

In order to compute the strength of each rule, first determine the rule's weight. Every rule can have a weight which can be a number between (0 and 1). The weight is applied to the single antecedent value that is achieved with the help of Fuzzy operators. Generally the rule weight is considered as 1, which does not affect the consequent part of the rule, but the rule weights can be changed by giving the different weight value (other than 1) to a rule relative to the other rule.

Step 8: Define Fuzzy Associative Memory

A Fuzzy Associative Memory is defined in the Matrix of Fuzzy values. It maps the input Fuzzy sets to the output Fuzzy sets. Which is stored in the knowledgebase of the Fuzzy Inference System.

Step 9: Find Consequences of the Rules

The consequent of each rule is a Fuzzy set represented by membership function obtained from implication process. The consequence of the rule is performed using an implication operator applied between the rule strength and the output membership function (output Fuzzy set). The implication is implemented for each rule. The implication operator clips the output membership function at the rule strength. The methods used for implication process are: *min* (minimum) which is also known as AND which truncates the output Fuzzy set, and the other method is *prod* (product), which scales the output Fuzzy set.

Step 10: Aggregate Rule Outputs

The decision of *FIS* is based on the testing of the all rules, therefore the consequents of all rules must be combined in order to make the decision. This is called the aggregation process which combines the output of each rule (Fuzzy set) into a single Fuzzy set. The aggregation process takes the input as the truncated output functions returned by the implication process of each rule. The normal aggregation methods used in Fuzzy Inference Systems are: *max* (maximum), *probor* (probabilistic OR) and *sum* (simply the sum of each rule's output set).

The aggregation of rule consequents is performed only for those rules which fall into the particular degrees of membership with respect to the crisp inputs among total number of possible set of rules. The minimum number of rules to be aggregated will be one rule if there is no overlapping between the degrees of memberships and the maximum number of rules involved into the aggregation will depend on the number of overlapping input crisp variable values.

Step 11: Defuzzify the output

Since the input given to the fuzzification step is in crisp values, while the rule consequents and aggregated output is the Fuzzy sets, therefore it is necessary to convert the output Fuzzy set into crisp value for final output. The Defuzzification step converts the aggregate of the output Fuzzy sets to a single crisp number. The most common defuzzification methods are *centroid* and *maximum*. The *centroid* method returns the center of area under the curve, while the maximum method returns the maximum truth for the output from the output Fuzzy sets in the form of crisp value. There are also some other methods used for defuzzification such as: middle of maximum, bisector, smallest of maximum and the largest of maximum etc.

4.4 QoS Calculation using Fuzzy Inference System

In this thesis the FIS model used for calculating the QoS is the Takagi-Sugeno-Kang method introduced by Sugeno [113]. The difference between the Sugeno method and Mamdani method is the difference of mechanism of output membership function, which is either constant or linear in the Sugeno method.

As already discussed in this chapter, the FIS systems are used to map inputs to outputs and the QoS calculation is the process of getting the QoS score as output based on different number of QoS terms as inputs. The total QoS score as compared to the final output of the FIS is the aggregation of various possible numbers of rules and their consequents based on the number of inputs. A rule consequent model diagram based on the number of inputs and one output using Takagi-Sugeno model is shown in Figure 4-3 and the aggregation of the number of rule consequents is shown in Figure 4-4.

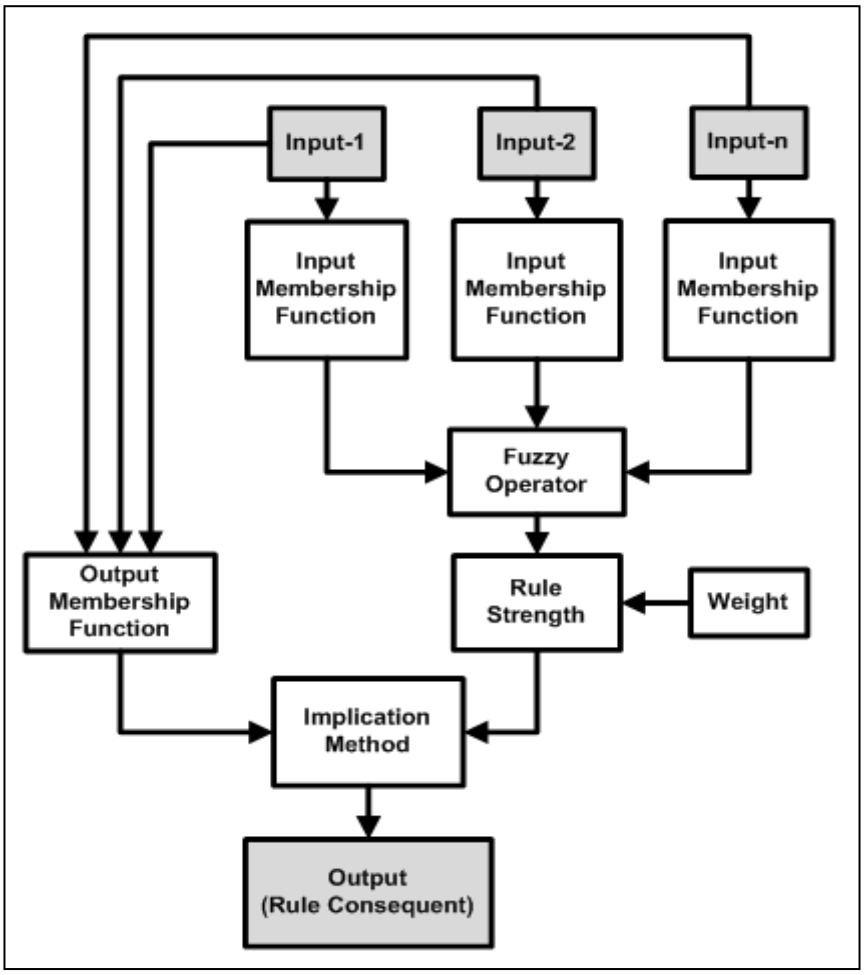

Figure 4-3: Block Diagram of Rule Consequent using Sugeno-Takagi Model

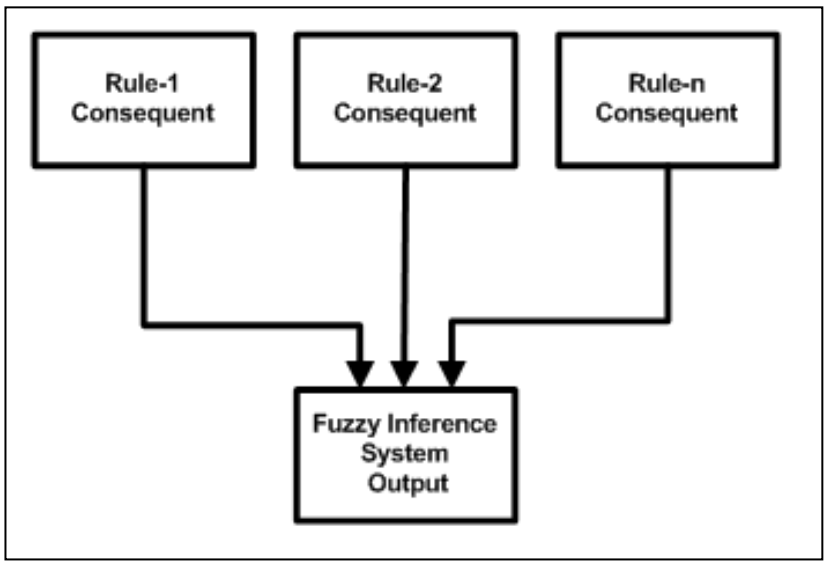

Figure 4-4: Fuzzy Inference System Output Diagram

The Fuzzy Inference Process using Sugeno Fuzzy model for QoS calculation is explained in detail using following steps:

Step 1: Determine the Number of Inputs and Outputs

A Fuzzy inference System can have one or more number of crisp inputs denoted by *X* is a set of *xn* individual X_i crisp inputs, and $1 \leq i \leq xn$.

$$X = \{X_1, X_2, \ldots, X_{xn}\} \text{ ---------- (Eq: 4-1)}$$

A Fuzzy inference System can have one or more number of final outputs denoted by *FO* is a set of *fon* individual FO_i final outputs, and $1 \leq i \leq fon$.

$$FO = \{FO_1, FO_2, \ldots, FO_{fon}\} \text{ ---------- (Eq:4-2)}$$

Step 2: Define Input Membership Functions

An input membership function denoted by *IMF* is a tuple of *X* crisp inputs set *(Eq:4-1)*, a set of Fuzzy membership value range as degree of membership for input denoted by *IR* and a set of Fuzzy linguistic variables for input membership function denoted by *ILV* (usually one or two words in each variable) corresponding to the degree or membership of Fuzzy set, where the range *IR* is $0 \leq IR \leq 1$, and *ILV* is *ilvn* individual ILV_i Fuzzy linguistic variables, and $1 \leq i \leq ilvn$.

$$IMF = (X, IR, ILV) \text{ ---------- (Eq: 4-3)}$$
$$IR = \{[0, 1]\} \text{ ---------- (Eq: 4-4)}$$
$$ILV = \{ILV_1, \ldots, ILV_{ilvn}\} \text{ ---------- (Eq: 4-5)}$$

Step 3: Define Output Membership Functions

An output membership function denoted by *OMF* is a tuple of *Z* polynomial equation function based on *X* crisp inputs *(Eq: 4-1)*, a set of Fuzzy output membership value range as degree of output membership function denoted by *OR* and a set of Fuzzy linguistic variables for output membership function denoted by *OLV* corresponding to

the degree or membership of Fuzzy set. Where the range *OR* is $0 \leq OR \leq 1$, and *OLV* is *ilvn* individual OLV_i Fuzzy linguistic variables, and $1 \leq i \leq olvn$.

The *Z* is usually a polynomial function used in every output membership function based on the input variable *X* with respect to the number of crisp input variables, where K_i is the coefficient variable of each *X* input variables in the antecedent of the rules and *j* is the constant for each rule.

$$OMF = (Z, OR, OLV) \text{ ---------- } (Eq: 4\text{-}6)$$

$$OR = \{[0, 1]\} \text{ ---------- } (Eq: 4\text{-}7)$$

$$OLV = \{OLV_1, ..., OLV_{olvn}\} \text{ ---------- } (Eq: 4\text{-}8)$$

$$Z = \{(K_i X_i +, ..., K_n X_n) + j\} \text{ ---------- } (Eq: 4\text{-}9)$$

A Sugeno Fuzzy model has the output level as constant when each coefficient variable K_i is equal to 0; this is called a zero-order Sugeno Model.

If each coefficient variable K_i in a *Z* polynomial equation is set to 1, then all input variables have equal scale of input, but if the scale of inputs is different then it can be adjusted by changing the value of coefficient variables to give a balance to Fuzzy Inference method.

Step 4: Determine the set of Fuzzy rules

Each Fuzzy rule *("If-Then")* has the two parts i.e. first part is called *antecedent* and the second part is called the *consequent*. The total number of Fuzzy rules denoted by *NR* is calculated as follows:

$$NR = [m]^n \text{ ---------- } (Eq: 4\text{-}10)$$

Where m is the number of linguistic membership variables in each input membership function *(Eq: 4-5)*, and n is the number of crisp inputs *(Eq: 4-1)* used in the Fuzzy Inference System. The out of total number of rules only certain number of rules will be

used in the Fuzzy Inference method depending on the overlapping value of each input crisp value with respect to the corresponding Fuzzy linguistic variable.

The maximum number of rules being used in the Aggregate Rule Output (Step-10) from total number of rules generated by *(Eq: 4-10)* is 2^k, where *k* is the total number of crisp inputs used in the Fuzzy Inference System. This also includes any number of rules being overlapped. However the minimum number of rules being applied for Aggregate Rule Output (Step-10) will be *k*, where k is the total number of crisp inputs.

Step 5: Fuzzify inputs

Each crisp input X_i is fuzzified with a corresponding input membership function *IMF* denoted by:

$$IMF(X) = IMF_i(X_i) \text{ ---------- (Eq: 4-11)}$$

Where *IMF* is input membership function *(Eq: 4-3)* and *X* is crisp input variable *(Eq: 4-1).*

Step 6: Combining the Fuzzified inputs

If more than one inputs are involved in the *antecedent* part of the rule (i.e. more than one crisp inputs), then after fuzzifying inputs *(Eq: 4-11)* they are converted to a single value using Fuzzy operator *FOP*.

$$FOP = \{AND, OR\} \text{ ---------- (Eq: 4-12)}$$

$$ANT = \{IMF(X_i)\ FOP\ IMF(X_{i+1}), .\ FOP\ .,\ IMF(X_{n-1})FOP\ IMF(X_n)\ \} \text{ ------- (Eq: 4-13)}$$

Where *FOP* is a set of Fuzzy operators and *ANT* is computed as the single value *antecedent* obtained by applying the *FOP* on more than one parts of the *antecedent* of the rule.

Step 7: Compute the Rule Strength

Each Fuzzy rule should have a rule strength, which is calculated by applying the number between (0 to 1) to the number given from antecedent of the rule is represented by:

RS= ANT (multiply) W---------- (Eq: 4-14)

Where Rule Strength is denoted by *RS*, weight of the rule between (0 to 1) is denoted by *W* is the, and single value of the antecedent part of the rule is denoted by ANT.

Step 8: Define Fuzzy Associative Memory

Fuzzy Associative Memory *(FAM)* is a table of total number of Fuzzy rules *(Eq: 4-10).* Table 4-1 represents the *antecedent* and *consequent* part of a rule using linguistic variables of Fuzzy sets from input *(Eq: 4-5)* and output *(Eq: 4-8)* membership functions. The *antecedent* part is joined by Fuzzy operator *FOP (Eq: 4-12)* and the *consequent* part is usually one variable represented by Fuzzy linguistic variable from output membership function *(Eq: 4-8).*

Table 4-1: Fuzzy Associative Memory

If	**More than two inputs**	**Then**
$IMF(ILV_i\ X_i)\ FOP\ IMF(ILV_i\ X_{(i+1)})$	$FOP\ (IMF\ (ILV_i\ X_{(n-1)}) FOP\ IMF(ILV_i\ X_n))$	$OMF(OLV_i)$

Where first column in Table 4-1 is the *antecedent* of Fuzzy rule which is based on two inputs with corresponding Fuzzy linguistic variables, middle column is continuation of *antecedent* part from first column to add more than two inputs and corresponding Fuzzy linguistic variables to get a single antecedent value. The third column is output Fuzzy linguistic variable from the consequent of the rule.

The useful number of rules for Fuzzy Inference Process from *FAM* depends on the value of crisp input variables that fall in the overlapping range of input values between the two Fuzzy linguistic variables ranges.

In order to formulate the FAM Table for deciding the outputs of the Fuzzy Inference System rules, the input Fuzzy linguistic variables i.e. Low, Medium and High are given numeric values 1, 2 and 3 respectively only for FAM Table creation purpose. The average numeric values based on two or three input Fuzzy linguistic variables helps to decide the output Fuzzy linguistic variable i.e. output of the rule in Fuzzy Associative Memory Table.

The all possible average calculations of numeric values given to Fuzzy linguistic variables based on two Inputs and its output Fuzzy linguistic variable with its value range is given in Table 4-2. The all possible average calculations of numeric values given to Fuzzy linguistic variables based on three Inputs and its output Fuzzy linguistic variable with its value range is given in Table 4-4.

The Fuzzy Associative Memory Table based on two inputs and one output with average values and its inferred output is given in Table 4-3, and Fuzzy Associative Memory Table based on three inputs and one output with average values and its inferred output is given in Table 4-5.

Table 4-2: Possible Average Values of Fuzzy Linguistic Variables for Two Inputs

POSSIBLE AVERAGE VALUES (INPUT1+INPUT2)/2	**LINGUISTIC OUTPUT**
>=1.0 AND <1.5	LOW (WEAK LOW) = 0
>=1.5 AND <2	LOW(STRONG LOW)= 1.665
>=2 AND <2.5	MEDIUM (WEAK MEDIUM)= 3.33
>=2.5 AND <3	MEDIUM(STRONG MEDIUM)=4.995
=3	HIGH(WEAK HIGH)= 6.66 HIGH (STRONG HIGH)= 8.325

Table 4-3: Fuzzy Associative Memory Table Based On Two Inputs

Input1	**Input2**	**Average Calculation**	**Output Fuzzy Linguistic Variable**
LOW	LOW	1.0	LOW (WEAK LOW) = 0
LOW	MEDIUM	1.5	LOW(STRONG LOW)= 1.665

LOW	HIGH	2	MEDIUM (WEAK MEDIUM)= 3.33
MEDIUM	LOW	1.5	LOW(STRONG LOW)= 1.665
MEDIUM	MEDIUM	2	MEDIUM (WEAK MEDIUM)= 3.33
MEDIUM	HIGH	2.5	MEDIUM(STRONG MEDIUM)=4.995
HIGH	LOW	2	MEDIUM (WEAK MEDIUM)= 3.33
HIGH	MEDIUM	2.5	MEDIUM(STRONG MEDIUM)=4.995
HIGH	HIGH	3	HIGH(WEAK HIGH)= 6.665 HIGH (STRONG HIGH)= 8.325

Table 4-4: Possible Average Values of Fuzzy Linguistic Variables for Three Inputs

POSSIBLE AVERAGE VALUES (INPUT1+INPUT2+INPUT3)/3	**LINGUISTIC OUTPUT**
>=1.0 AND <1.33	LOW (WEAK LOW) = 0
>=1.33 AND <1.66	LOW(STRONG LOW)= 1.665
>=1.667 AND <2	MEDIUM (WEAK MEDIUM)= 3.33
>=2 AND <2.33	MEDIUM(STRONG MEDIUM)=4.995
>=2.33 AND <2.667	HIGH(WEAK HIGH)= 6.665
>=2.667 AND <=3	HIGH (STRONG HIGH)= 8.325

Table 4-5: Fuzzy Associative Memory Table Based On Three Inputs

Input1	**Input2**	**Input3**	**Average Calculation**	**Output Fuzzy Linguistic Variable**
LOW	LOW	LOW	1	LOW (WEAK LOW) = 0
LOW	LOW	MEDIUM	1.33	LOW(STRONG LOW)= 1.665
LOW	LOW	HIGH	1.667	MEDIUM (WEAK MEDIUM)= 3.33
LOW	MEDIUM	LOW	1.33	LOW(STRONG LOW)= 1.665
LOW	MEDIUM	MEDIUM	1.667	MEDIUM (WEAK MEDIUM)= 3.33
LOW	MEDIUM	HIGH	2	MEDIUM(STRONG MEDIUM)=4.995
LOW	HIGH	LOW	1.667	MEDIUM (WEAK MEDIUM)= 3.33
LOW	HIGH	MEDIUM	2	MEDIUM(STRONG MEDIUM)=4.995
LOW	HIGH	HIGH	2.33	HIGH(WEAK HIGH)= 6.665
MEDIUM	LOW	LOW	1.33	LOW(STRONG LOW)= 1.665
MEDIUM	LOW	MEDIUM	1.667	MEDIUM (WEAK MEDIUM)= 3.33
MEDIUM	LOW	HIGH	2	MEDIUM(STRONG MEDIUM)=4.995
MEDIUM	MEDIUM	LOW	1.667	MEDIUM (WEAK MEDIUM)= 3.33

MEDIUM	MEDIUM	MEDIUM	2	MEDIUM(STRONG MEDIUM)=4.995
MEDIUM	MEDIUM	HIGH	2.33	HIGH(WEAK HIGH)= 6.665
MEDIUM	HIGH	LOW	2	MEDIUM(STRONG MEDIUM)=4.995
MEDIUM	HIGH	MEDIUM	2.33	HIGH(WEAK HIGH)= 6.665
MEDIUM	HIGH	HIGH	2.667	HIGH (STRONG HIGH)= 8.325
HIGH	LOW	LOW	1.667	MEDIUM (WEAK MEDIUM)= 3.33
HIGH	LOW	MEDIUM	2	MEDIUM(STRONG MEDIUM)=4.995
HIGH	LOW	HIGH	2.33	HIGH(WEAK HIGH)= 6.665
HIGH	MEDIUM	LOW	2	MEDIUM(STRONG MEDIUM)=4.995
HIGH	MEDIUM	MEDIUM	2.33	HIGH(WEAK HIGH)= 6.665
HIGH	MEDIUM	HIGH	2.667	HIGH (STRONG HIGH)= 8.325
HIGH	HIGH	LOW	2.33	HIGH(WEAK HIGH)= 6.665
HIGH	HIGH	MEDIUM	2.667	HIGH (STRONG HIGH)= 8.325
HIGH	HIGH	HIGH	3	HIGH (STRONG HIGH)= 8.325

Step 9: Find Consequences of the Rules

The consequent parts of rules denoted by *CONS* is a set of *consn* individual rule consequences. Each consequence of rule $CONS_i$ is obtained by clipping the maximum degree of output Membership value at the rule strength RS_i value using implication operator *IMOP* is:

$$IMOP = \{AND, PROD\} \text{----------} \quad (Eq: 4\text{-}15)$$

$$CONS = \{CONS_1, \ldots, CONS_{consn}\} \text{----------} \quad (Eq: 4\text{-}16)$$

$$CONS_i = OMF_i\,(IMOP)\,RS_i \text{----------} \quad (Eq: 4\text{-}17)$$

Where the *OMF* is the output membership function *(Eq: 4-6)*, implication operator *IMOP (Eq: 4-15)* used is *AND*, and RS_i is the rule strength of each rule *(Eq: 4-14)*. The implication operator clips the output membership function value at rule strength to obtain the rule consequent.

Step 10: Aggregate Rule Outputs

The aggregation of the rules outputs denoted by *AGR* is the summation of the all n number of rules consequents, which is given below:

$$AGR = \sum_{i=1}^{n} CONSi$$ ---------- *(Eq: 4-18)*

Step 11: Defuzzify output

The final output of the Fuzzy Inference System denoted by *FO* is given below:

$$FO = \frac{1}{n} \sum_{i=1}^{n} CONSi$$ ---------- *(Eq: 4-19)*

Where the FO is the average of *n* number of applicable rule consequents.

4.5 Service Composition Based on QoS

Composite service plans for Cloud computing services are formed on the basis of QoS scores of individual services. The QoS terms as metrics defined for QoS calculation are taken from Table 2-43. Any other domain specific QoS terms as a metric can also be used in the QoS calculation model of this thesis. The QoS score of a service can depend on one or more number of QoS Term metrics and the aggregation of QoS scores from different metrics is accumulated using Fuzzy Inference Process in this study. The composition of different services for consumer requirements can also be accumulated using the normal aggregation of QoS scores from two or more services in order to make a composite service plan decision.

The ranking of service providers is calculated by obtaining the Quality of Service score, which is actually calculated from rating values assigned to each participating QoS Term metric for QoS calculation with the help of feedback using monitoring tools depending on the delegation of role which defines how to get the feedback or QoS score value. The values for QoS Term metrics can be obtained by using Cloud monitoring tools from Table 2-27. In this thesis, the QoS terms as metric for calculation have been assigned the score value ranging (0 to 10) which is shown in Table 4-6, where the value 0 is considered as the low score and value 10 is considered as the high score.

Table 4-6: QoS Terms value Range

Category of Terms	QoS Terms	QoS Value Range
Service Provider Factors	1. Life Span	0-10
	2. Financial Status	0-10
	3. Branches	0-10
	4. Employees Of Organization	0-10
	5. No Of Services	0-10
	6. Brand Value	0-10
	7. Success Rate	0-10
	8. Advertising	0-10
Trust	9. Access Trust	0-10
	10. Provision Trust	0-10
	11. Certification of Trust	0-10
	12. Delegation Trust	0-10
	13. Infrastructure Trust	0-10
Performance	14. Throughput	0-10
	15. Response Time	0-10
	16. Latency	0-10
	17. Execution Time	0-10
	18. Transaction Time	0-10
Security	19. Authentication	0-10
	20. Authorization	0-10
	21. Accountability	0-10
	22. Confidentiality	0-10
	23. Traceability and Auditability	0-10
	24. Non-Repudiation	0-10
	25. Encryption	0-10
Dependability	26. Availability	0-10
	27. Accessibility	0-10
	28. Accuracy	0-10
	29. Reliability	0-10
	30. Capacity	0-10
	31. Scalability	0-10
	32. Exception Handling (Stability)	0-10
	33. Robustness (Flexibility)	0-10
	34. Integrity (Data and Transaction)	0-10
Domain Specific Terms	35. Any Attribute	0-10

4.6 Mockup Calculations for QoS using FIS

The QoS calculation method using Fuzzy Inference System in this thesis can be applied to the QoS terms defined in Table 2-33 and Table 2-42. The QoS calculation diagram is show in Figure 4-5.

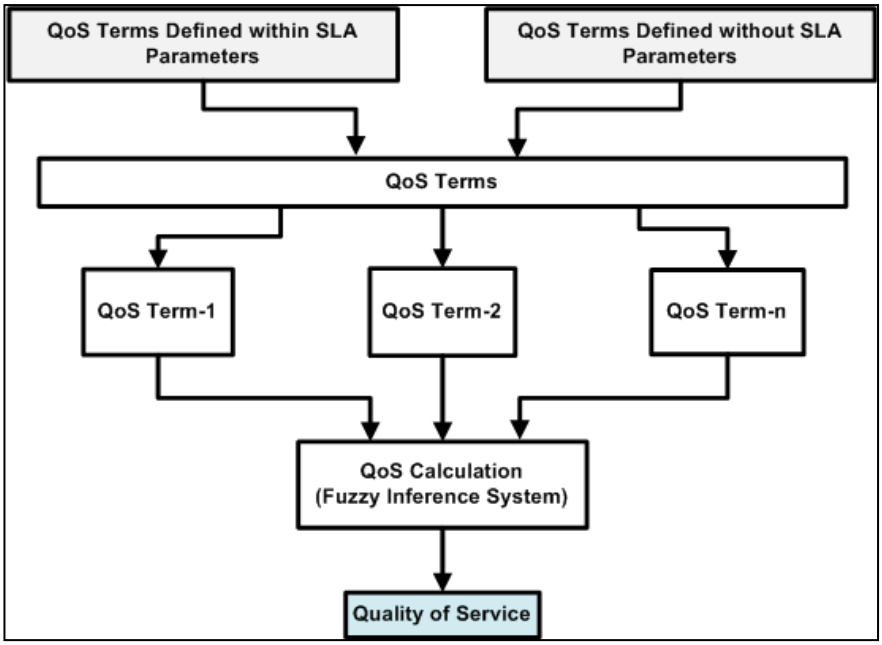

Figure 4-5: QoS Calculation Diagram

4.6.1 Steps for QoS Calculation

In order to calculate the QoS score, the recommended steps of Fuzzy Inference Process mentioned in Section 4.4 are performed for the mockup example with requirements given in Table 4-7 for two inputs and one output and in Table 4-8 the requirements are given for three inputs and one output.

The mockup example calculations based on two inputs and one output using Fuzzy Inference Process (Step 1 to Step 11) are shown in Table 4-9.

The mockup calculations based on three inputs and one output using Fuzzy Inference Process (step 1 to step 11) are shown in Table 4-10 in short, while the full list of FIS calculations is shown in the Appendix-B, Table 10-3.

The Fuzzy Inference Process implementation code using Excel VBA programming based on two inputs and one output is given in Appendix-B, Table 10-1. The Fuzzy Inference Process implementation code based on three inputs and one output is given in Appendix-B, Table 10-2.

Table 4-7: Mockup Example Requirements for Two Inputs and One Output

S.No	Requirements	Values
1	Number of QoS Input Parameters	2
2	Names of QoS Input Parameters	X1, X2
3	Number of Input Membership Functions (IMF)	2
4	Number of Output Membership Functions (OMF)	1
5	Number of Fuzzy Linguistic Variables in each IMF (ILV)	3
6	Names of Fuzzy Linguistic Variables in IMF (ILV)	Low, Medium, High
7	Number of Fuzzy Linguistic Variables in each OMF (OLV)	6
8	Names of Fuzzy Linguistic Variables in OMF (OLV)	Weak Low, Strong Low, Weak Medium, Strong Medium, Weak High, Strong High
9	Ranges of Fuzzy Linguistic Variables in OMF (OR)	Low (0 to 3.33), Medium (3.33 to 6.66), High (6.66 to 10), while each Low, Medium and High is further divided into Weak Low (0 to 1.665,), Strong Low (1.665 to 3.33), Weak Medium (3.33 to 4.995), Strong Medium (4.995 to 6.66), Weak High (6.66 to 8.325), Strong High (8.325 to 10) respectively shown in Figure 4-7 to Figure 4-12.
10	Range of Fuzzy Linguistic Variables in IMF (IR)	Low (0 to 4), Medium (3 to 7), High (6 to 10) shown in Figure 4-6.
11	Number of Fuzzy Rules	$[3]^2 = 9$ according to *(Eq: 4-10)*

Table 4-8 : Mockup Example Requirements for Three Inputs and One Output

S.No	Requirements	Values
1	Number of QoS Input Parameters	3
2	Names of QoS Input Parameters	X1, X2, X3
3	Number of Input Membership Functions (IMF)	3
4	Number of Output Membership Functions (OMF)	1
5	Number of Fuzzy Linguistic Variables in each IMF (ILV)	3
6	Names of Fuzzy Linguistic Variables in IMF (ILV)	Low, Medium, High
7	Number of Fuzzy Linguistic Variables in each OMF (OLV)	6
8	Names of Fuzzy Linguistic Variables in OMF (OLV)	Weak Low, Strong Low, Weak Medium, Strong Medium, Weak High, Strong High
9	Ranges of Fuzzy Linguistic Variables in OMF (OR)	Low (0 to 3.33), Medium (3.33 to 6.66), High (6.66 to 10), while each Low, Medium and High is further divided into Weak Low (0 to 1.665,), Strong Low (1.665 to 3.33), Weak Medium (3.33 to 4.995), Strong Medium (4.995 to 6.66), Weak High (6.66 to 8.325), Strong High (8.325 to 10) respectively shown in Figure 4-7 to Figure 4-12.
10	Range of Fuzzy Linguistic Variables in IMF (IR)	Low (0 to 4), Medium (3 to 7), High (6 to 10) shown in Figure 4-6.
11	Number of Fuzzy Rules	$[3]^3 = 27$

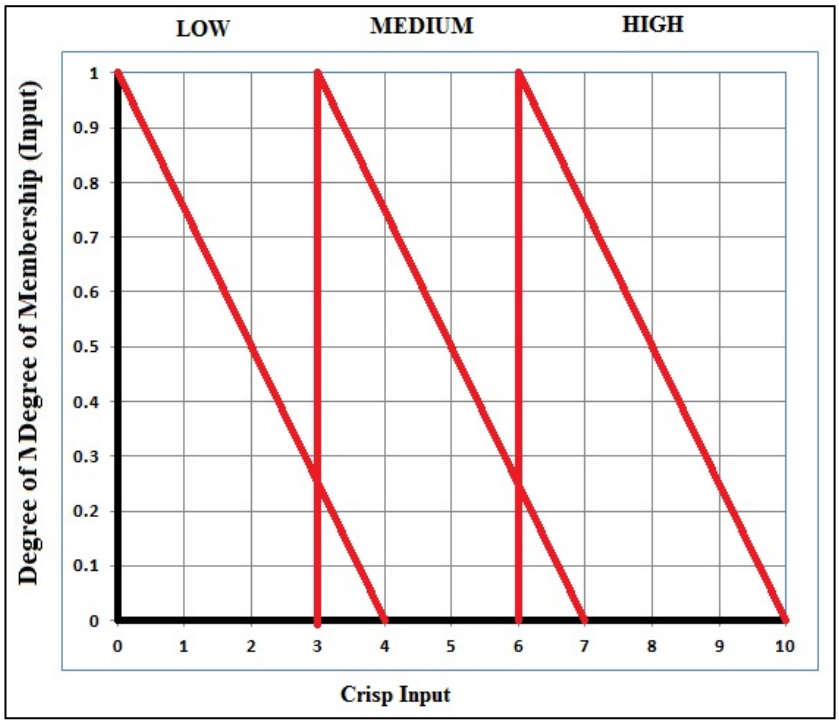

Figure 4-6: Input Membership function graph

In Figure 4-6, X-Axis is used for crisp input, and Y-Axis is used for degree of membership for each input Fuzzy linguistic variable and following equations can be used to get the degrees of input membership function:

In the Figure 4-7 to Figure 4-12, the X-Axis is used for crisp output, and Y-Axis is used for the degree of membership for each output Fuzzy linguistic variable for degree of output membership function.

A. Equations for LOW membership

a. If X = 0 then

Y=1 ---------- *(Eq: 4-20)*

b. If 0>X<=4 and (X_1=0, Y_1= 1)(X_2=4, Y_2= 0) then

$$\frac{y-y1}{y2-y1} = \frac{x-x1}{x2-x1}$$

$$\frac{y-1}{0-1} = \frac{x-0}{4-0}$$

$$\frac{y-1}{-1} = \frac{x}{4}$$

$$y = \frac{-x}{4} + 1$$ ---------- *(Eq: 4-21)*

B. Equations for MEDIUM Memberships

a. If X=3 then

Y=1 ---------- *(Eq: 4-22)*

b. If 3>X<=7 and (X_1=3, Y_1=1)(X_2=7, Y_2=0) then

$$\frac{y-y1}{y2-y1} = \frac{x-x1}{x2-x1}$$

$$\frac{y-1}{0-1} = \frac{x-3}{7-3}$$

$$y = \frac{-x+3}{4}$$

$$y = \frac{-x+3}{4} + 1$$ ---------- *(Eq: 4-23)*

C. Equations for HIGH Memberships

a. If X=6 then

Y=1 ---------- *(Eq: 4-24)*

b. If 6>X<=10 and (X_1=6, Y_1= 1)(X_2= 10, Y_2= 0) then

$$\frac{y-y1}{y2-y1} = \frac{x-x1}{x2-x1}$$

$$\frac{y-1}{0-1} = \frac{x-6}{10-6}$$

$$\frac{y-1}{-1} = \frac{x-6}{4}$$

$$y - 1 = \frac{-x+6}{4}$$

$$y = \frac{-x+6}{4} + 1$$ ---------- *(Eq: 4-25)*

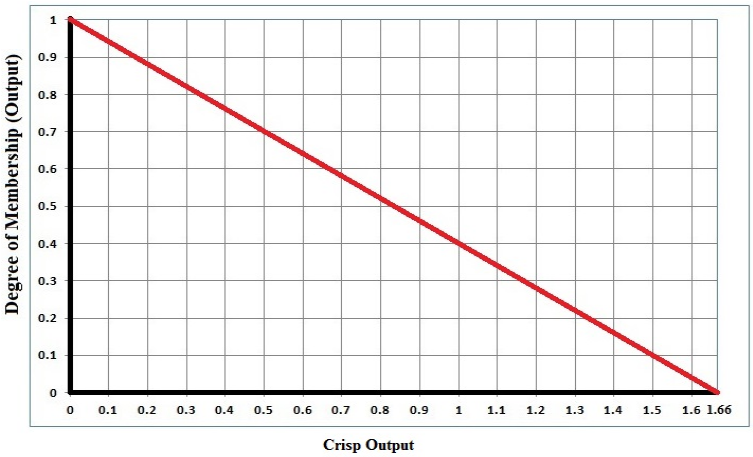

Figure 4-7: Weak-Low Output Membership Function Graph

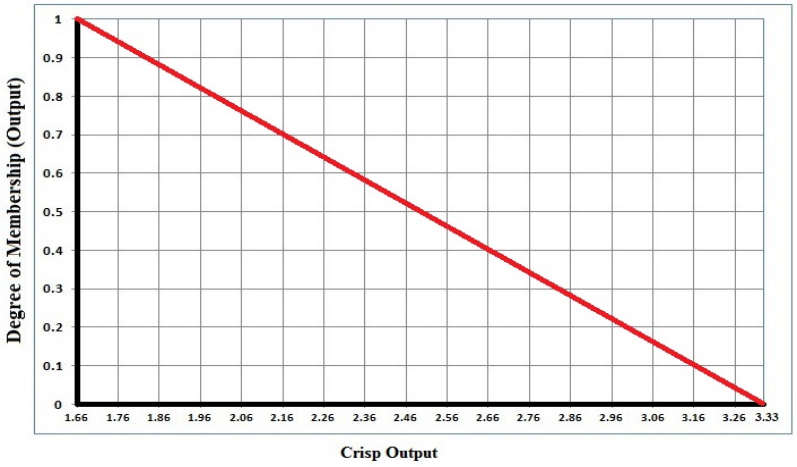

Figure 4-8: Strong-Low Output Membership Function Graph

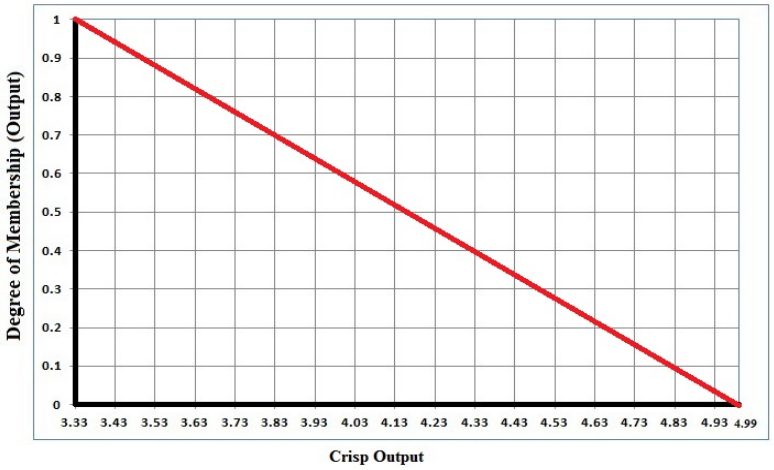

Figure 4-9: Weak-Medium Output Membership Function Graph

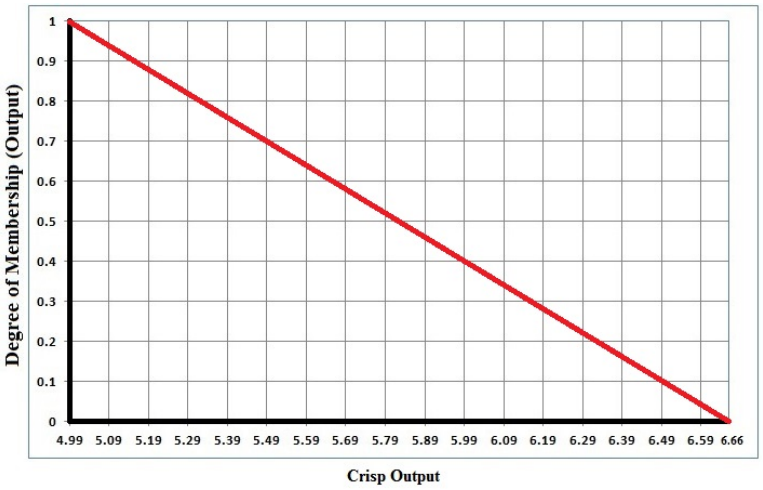

Figure 4-10: Strong-Medium Output Membership Function Graph

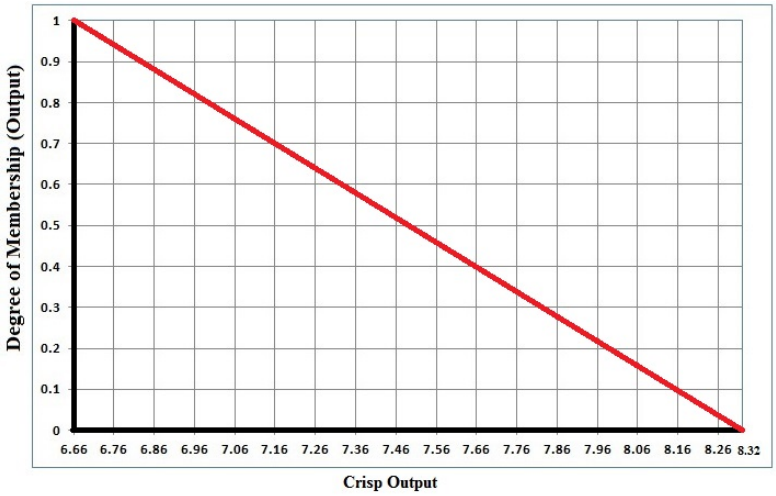

Figure 4-11: Weak-High Output Membership Function Graph

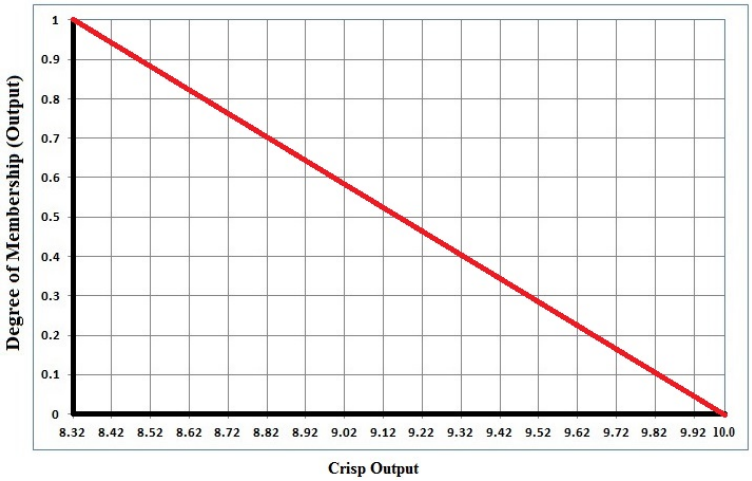

Figure 4-12: Strong- High Output Membership Function Graph

Table 4-9: FIS Mockup Calculations based on Two Inputs and One Output

S. No	Input1	Input2	Output	S. No	Input1	Input2	Output	S. No	Input 1	Input2	Output
1	10	0	3.33	**38**	7	4	4.058438	**75**	4	8	4.786875
2	10	1	3.74625	**39**	7	5	4.786875	**76**	4	9	4.995
3	10	2	4.1625	**40**	7	6	5.8275	**77**	4	10	5.203125
4	10	3	4.786875	**41**	7	7	6.139687	**78**	3	0	0.8325
5	10	4	5.203125	**42**	7	8	7.284375	**79**	3	1	1.040625
6	10	5	5.8275	**43**	7	9	7.4925	**80**	3	2	1.24875
7	10	6	7.284375	**44**	7	10	7.700625	**81**	3	3	1.977188
8	10	7	7.700625	**45**	6	0	2.4975	**82**	3	4	2.08125
9	10	8	9.1575	**46**	6	1	2.705625	**83**	3	5	2.91375
10	10	9	9.57375	**47**	6	2	2.91375	**84**	3	6	3.642188
11	10	10	9.99	**48**	6	3	3.642188	**85**	3	7	3.74625
12	9	0	3.33	**49**	6	4	3.74625	**86**	3	8	4.57875
13	9	1	3.74625	**50**	6	5	4.57875	**87**	3	9	4.786875
14	9	2	4.1625	**51**	6	6	5.723438	**88**	3	10	4.786875
15	9	3	4.786875	**52**	6	7	5.8275	**89**	2	0	0
16	9	4	4.995	**53**	6	8	7.07625	**90**	2	1	0.41625
17	9	5	5.8275	**54**	6	9	7.284375	**91**	2	2	0.8325
18	9	6	7.284375	**55**	6	10	7.284375	**92**	2	3	1.24875
19	9	7	7.4925	**56**	5	0	1.665	**93**	2	4	1.456875
20	9	8	9.1575	**57**	5	1	2.08125	**94**	2	5	2.4975
21	9	9	9.57375	**58**	5	2	2.4975	**95**	2	6	2.91375
22	9	10	9.57375	**59**	5	3	2.91375	**96**	2	7	3.121875
23	8	0	3.33	**60**	5	4	3.121875	**97**	2	8	4.1625
24	8	1	3.74625	**61**	5	5	4.1625	**98**	2	9	4.1625
25	8	2	4.1625	**62**	5	6	4.57875	**99**	2	10	4.1625
26	8	3	4.57875	**63**	5	7	4.786875	**100**	1	0	0
27	8	4	4.786875	**64**	5	8	5.8275	**101**	1	1	0.41625
28	8	5	5.8275	**65**	5	9	5.8275	**102**	1	2	0.41625
29	8	6	7.07625	**66**	5	10	5.8275	**103**	1	3	1.040625
30	8	7	7.284375	**67**	4	0	0.8325	**104**	1	4	1.24875
31	8	8	9.1575	**68**	4	1	1.24875	**105**	1	5	2.08125
32	8	9	9.1575	**69**	4	2	1.456875	**106**	1	6	2.705625
33	8	10	9.1575	**70**	4	3	2.08125	**107**	1	7	2.91375
34	7	0	2.4975	**71**	4	4	2.393438	**108**	1	8	3.74625
35	7	1	2.91375	**72**	4	5	3.121875	**109**	1	9	3.74625
36	7	2	3.121875	**73**	4	6	3.74625	**110**	1	10	3.74625
37	7	3	3.74625	**74**	4	7	4.058438				

Table 4-10: FIS Mockup Calculations based on Three Inputs and One Output

S.No	Input1	Input2	Input3	FIS Output	S.No	Input1	Input2	Input3	FIS Output
1	10	10	0	4.995	670	5	5	9	5.8275
10	10	10	9	9.57375	680	5	4	8	4.786875
20	10	9	8	9.1575	690	5	3	7	3.6421875
30	10	8	7	8.119375	700	5	2	6	2.91375
40	10	7	6	7.07875	710	5	1	5	2.08125
50	10	6	5	6.24625	720	5	0	4	1.665
60	10	5	4	4.786875	730	4	10	3	3.74625
70	10	4	3	3.74625	740	4	9	2	3.121875
80	10	3	2	2.91375	750	4	8	1	2.91375
90	10	2	1	2.08125	760	4	7	0	2.08125
100	10	1	0	1.665	770	4	7	10	5.7246875
110	10	1	10	5.41125	780	4	6	9	5.4125
120	10	0	9	4.995	790	4	5	8	4.786875
130	9	10	8	9.1575	800	4	4	7	3.27796875
140	9	9	7	8.3275	810	4	3	6	2.91375
150	9	8	6	7.91125	820	4	2	5	2.289375
160	9	7	5	6.454375	830	4	1	4	1.665
170	9	6	4	5.4125	840	4	0	3	1.24875
180	9	5	3	4.57875	850	3	10	2	2.91375
190	9	4	2	3.121875	860	3	9	1	2.705625
200	9	3	1	2.705625	870	3	8	0	2.4975
210	9	2	0	1.665	880	3	8	10	6.24625
220	9	2	10	5.8275	890	3	7	9	5.4125
230	9	1	9	5.41125	900	3	6	8	5.204375
240	9	0	8	4.995	910	3	5	7	3.6421875
250	8	10	7	8.119375	920	3	4	6	2.91375
260	8	9	6	7.91125	930	3	3	5	2.289375
270	8	8	5	7.4975	940	3	2	4	1.5609375
280	8	7	4	5.5165625	950	3	1	3	1.3528125
290	8	6	3	5.204375	960	3	0	2	0.8325
300	8	5	2	4.1625	970	2	10	1	2.08125
310	8	4	1	2.91375	980	2	9	0	1.665
320	8	3	0	2.4975	990	2	9	10	5.8275
330	8	3	10	6.24625	1000	2	8	9	5.8275
340	8	2	9	5.8275	1010	2	7	8	4.786875
350	8	1	8	5.41125	1020	2	6	7	3.6421875
360	8	0	7	4.1625	1030	2	5	6	2.91375
370	7	10	6	7.07875	1040	2	4	5	2.289375
380	7	9	5	6.454375	1050	2	3	4	1.5609375
390	7	8	4	5.5165625	1060	2	2	3	1.24875
400	7	7	3	4.4753125	1070	2	1	2	0.41625
410	7	6	2	3.6421875	1080	2	0	1	0
420	7	5	1	2.91375	1090	1	10	0	1.665
430	7	4	0	2.08125	1100	1	10	10	5.41125

440	7	4	10	5.7246875	1110	1	9	9	5.41125
450	7	3	9	5.4125	1120	1	8	8	5.41125
460	7	2	8	4.786875	1130	1	7	7	3.74625
470	7	1	7	3.74625	1140	1	6	6	3.4340625
480	7	0	6	3.33	1150	1	5	5	2.08125
490	6	10	5	6.24625	1160	1	4	4	1.665
500	6	9	4	5.4125	1170	1	3	3	1.3528125
510	6	8	3	5.204375	1180	1	2	2	0.41625
520	6	7	2	3.6421875	1190	1	1	1	0.41625
530	6	6	1	3.4340625	1200	1	0	0	0
540	6	5	0	2.4975	1210	1	0	10	1.665
550	6	5	10	6.24625	1220	0	10	9	4.995
560	6	4	9	5.4125	1230	0	9	8	4.995
570	6	3	8	5.204375	1240	0	8	7	4.1625
580	6	2	7	3.6421875	1250	0	7	6	3.33
590	6	1	6	3.4340625	1260	0	6	5	2.4975
600	6	0	5	2.4975	1270	0	5	4	1.665
610	5	10	4	4.786875	1280	0	4	3	1.24875
620	5	9	3	4.57875	1290	0	3	2	0.8325
630	5	8	2	4.1625	1300	0	2	1	0
640	5	7	1	2.91375	1310	0	1	0	0
650	5	6	0	2.4975	1320	0	1	10	1.665
660	5	6	10	6.24625	1330	0	0	9	1.665

4.7 Chapter Summary

This chapter introduced the Fuzzy Inference Systems and its use for Quality of Service calculations for Cloud computing services on the basis of QoS terms as metrics defined within and without SLAs. A detailed Fuzzy Inference Process was explained for understanding and implementation of QoS calculations. The ideas for composition of services on the basis of QoS score was discussed in this chapter along with the selection of ranges for QoS terms as metrics for the QoS values. Finally, mockup calculations tables were generated for two inputs and three inputs to test the Fuzzy Inference Process for QoS calculation.

5. Composition Using SLAs

5.1 Introduction

The use of Service Oriented Architecture (SOA) helps to create loosely coupled distributed systems, composition of services, reduces cost of the services, interoperable and scalable systems within heterogeneous environments. However, on the large scale the SOA based systems lead to the problems of service discovery, service selection and management of services. It also becomes a challenge to determine the Quality of Services (QoS) for large number of services.

When one service is not sufficient to fulfill the requirements of the consumer then more than one service can be combined to form a composite service to accomplish the consumer requirements. The composition will be based on the best choices of different services selected according to Quality of Service using SLAs. The composition types used can be static or dynamic. The static or dynamic composition of services requires contracts to be agreed between the service providers and service consumers.

This study extends the SOA Triangle which is comprised of service consumer, service provider and UDDI by adding another component known as the SLAAgent for creating the flexible and efficient communication between the existing components of the SOA triangle which is shown in Figure 5-1. The SLAAgent incorporates the refined SLA elements from Chapter 2 (Table 2-6), SLA lifecycle stages from (Table 2-20), QoS terms from (Table 2-43) and Fuzzy Inference System process steps for QoS calculation from Chapter 4 (Section 4.4). The SLAAgent mainly focuses on the use of SLAs for determining the Quality of Service, service discovery, service selection, service composition, monitoring and management of services dynamically.

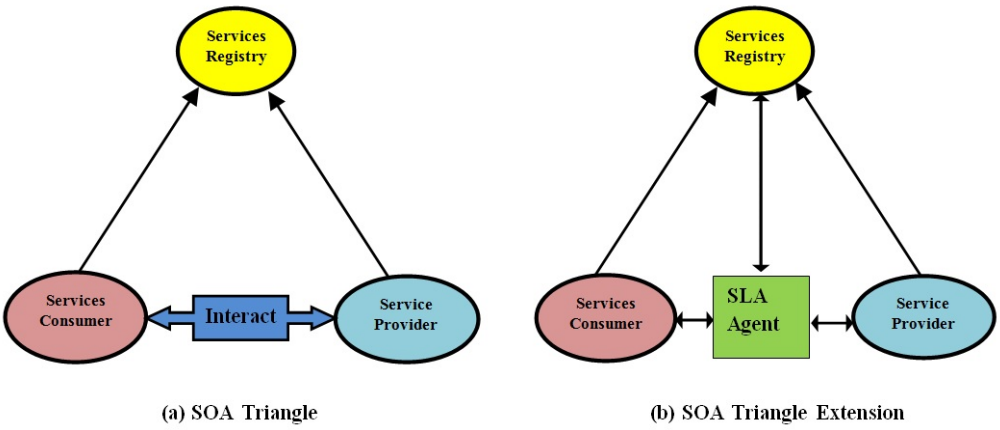

Figure 5-1: SOA Triangle and SOA Triangle Extension

5.2 High Level Framework Components

The proposed framework is based on the extension of the SOA Triangle, where an additional component called an SLAAgent is introduced. The SLAs in the SLAAgent are used for creating the bridge for composition as well as monitoring the Quality of Service. The SLAAgent has six major components including Requirement Processor (RP), Service Manager (SM), SLA Manager (SLAM), Quality of Service Monitor (QoSM), Quality of Service Database (QoSD) and Controller (C) that are shown in Figure 5-2.

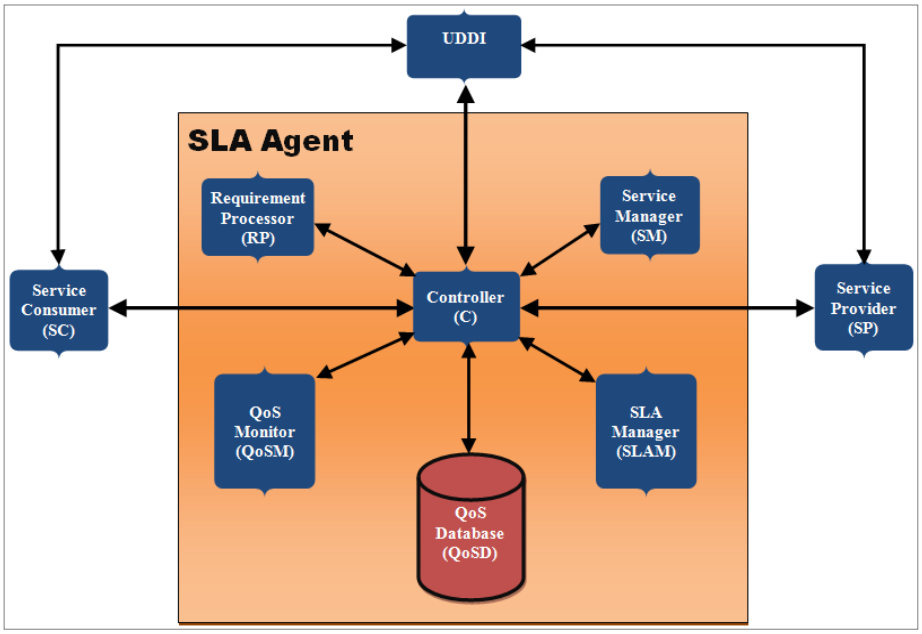

Figure 5-2 : High Level Framework Components

Controller (C): The controller is responsible for managing the communication between service providers, service consumers and registry of services. All SLAAgent components communicate to each other via the Controller. The Controller is the most active component of the framework which remains active all the time for sending, receiving and scheduling any flow of control actions for the other components of the framework.

Requirement Processor (RP): In this component, the consumer requirements are received and refined according to the service categories required to the consumers. The SLAs, services, service providers and their QoS information are requested by this component from service providers and the QoS Database component.

Service Manager (SM): This component is responsible for dealing with services and service providers for requesting to make and prepare the services for execution in single or composite group of services, and finally performs decommission if necessary.

SLA Manager (SLAM): This component is responsible for creating the SLA structures and management of SLAs during and after service provision. This component also contains a sub component to deals with negotiation between the service consumer and service provider in order to come to a mutual agreement for services. In this component the responsibilities of the service providers and service consumers are managed till the completion of services. It also includes the post service provision actions.

QoS Monitor (QoSM): This component is responsible for monitoring the services and SLAs, calculating the QoS based on different metric terms using Fuzzy Inference System proposed in Chapter 4.

QoS Database (QoSD): This component stores the individual QoS score of different services from service providers.

5.3 Low Level Framework Components

Each framework component has sub-components which are shown in Figure 5-3 and these sub-components are described in detail below:

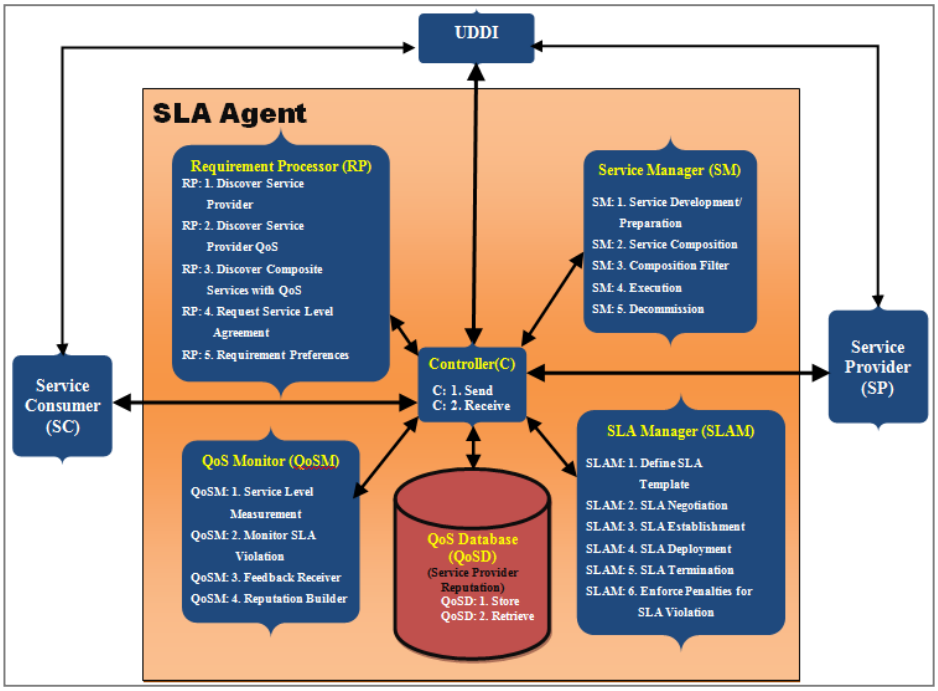

Figure 5-3 : Low Level Framework Components

5.3.1 Controller (C)

5.3.1.1 Send (C:1)

This sub-component of the Controller is used for sending any message to any component in the framework.

5.3.1.2 Receive (C:2)

This sub-component of the Controller is responsible for receiving any message or reply from any framework component.

5.3.2 Requirement Processor (RP)

5.3.2.1 Discover Service Provider (RP:1)

When requirements are submitted from a service consumer, this sub-component searches for the required services from the Services Registry (UDDI).

5.3.2.2 Discover Service Provider QoS (RP:2)

Once the relevant services are found from the services Registry (UDDI), this sub-component searches the Quality of Service from the QoS Database and retrieves the QoS of service score for the all required services.

5.3.2.3 Discover Composite Services with QoS (RP:3)

When a service consumer requires more than one service, this sub-component sends request to service composition sub-component of Service Manager (SM) for producing a composite services plan along with QoS information.

5.3.2.4 Request Service Level Agreement (RP:4)

If the required services are found, the service consumer can request the Service Level Agreement for each service for more details about the service offers defined in the SLAs from service providers.

5.3.2.5 Requirement Preferences (RP:5)

The composite services plan required to consumer with QoS information can result in many possible composite service combinations, while in this sub-component, the service consumer provides the requirement preferences for the services to narrow down the selection for most suitable service plan.

5.3.3 Services Manager (SM)

5.3.3.1 Service Development/ Preparation (SM:1)

If the normal predefined service offers are not sufficient for consumer requirements, then based on specific consumer request the service reconfiguration can be requested from the service providers from this sub-component.

5.3.3.2 Service Composition (SM:2)

If service consumer requires more than one service in a composite form of services in order to fulfill the collective requirements, this sub-component composes the services and makes the number of service composition options. More than one composite solutions are offered to the service consumer so that consumer can select most suitable

combination and buy any one of them according to the requirement preference based on QoS information about the service providers.

5.3.3.3 Composition Filter (SM:3)

Based on the requirement preferences from consumer defined in (RP:5), this sub-component filters the outputs of service composition plans (SM:2), to the selected number of composite service plans for easy decision making by the consumer.

5.3.3.4 Execution (SM:4)

The delivery of services is triggered by this sub-component for service consumers once the service consumer and service provider agree the on terms and conditions of the services that are explicitly defined in SLA between the both parties.

5.3.3.5 Decommission (SM:5)

Whenever a particular service or number of services and their communication is no longer needed, this sub-component stops all functionalities and communications of services safely.

5.3.4 SLA Manager (SLAM)

5.3.4.1 Define SLA Template (SLAM:1)

Once service providers have been found and a service consumer wants to get a service or number of services, this component extracts the elements of the SLAs provided from the service providers for the required service and then this component restructures them into a formal ontological shape of SLA compatible to the SLAAgent framework.

5.3.4.2 SLA Negotiation (SLAM:2)

This sub-component provides the exchange of messages for service SLA in the form of offers and counter offers between service consumer and service provider in order to reach a mutual agreement.

5.3.4.3 SLA Establishment (SLAM:3)

This sub-component performs the accomplishment of SLA formation process followed by digitally accepted or signed by signatory parties of the agreement.

5.3.4.4 SLA Deployment (SLAM:4)

This sub-component distributes the SLA between the involved parties and validates established SLA by sending the information contained in the SLA to the involving parties.

5.3.4.5 SLA Termination (SLAM:5)

This sub-component will terminate the SLA according to the predefined duration of the SLA or under the conditions which force the termination of the SLA due to situations such as violation of SLA by service provider or service consumer.

5.3.4.6 Enforce Penalties for SLA Violation (SLAM:6)

This sub-component will enforce the penalties for SLA Violation which are predefined in an SLA on behalf of the service providers if they fail to provide the adequate service which comply with Service Level Objectives (SLOs).

5.3.5 QoS Monitor (QoSM)

5.3.5.1 Service Level Measurement (QoSM:1)

This sub-component retains the current configuration of services defined in the SLAs and keeps checking those configurations at runtime.

5.3.5.2 Monitor SLA Violation (QoSM:2)

This sub-component compares the obligations defined in the SLA and actual service delivered to the consumer. The violation from consumer side also monitored and the accountable party will be dealt according to the obligations defined in SLA or re-alignment of the service may be performed according to the SLA.

5.3.5.3 Feedback Receiver (QoSM:3)

This sub-component is responsible for receiving the feedback from monitoring tools about the functionality of service at the end of SLA completion on normal time, or the abnormal termination due to any reason.

5.3.5.4 Reputation Builder (QoSM:4)

This sub-component accumulates the reputation of service provider as QoS score based on the feedback score received from monitoring tools about the service used with the help of Fuzzy Inference Methods defined in this thesis.

5.3.6 QoS Database (QoSD)

5.3.6.1 Store (QoSD:1)

This sub-component stores the service provider QoS score generated from Reputation Builder (QoSM: 4), in a persistent storage for future use.

5.3.6.2 Retrieve (QoSD:2)

This sub-component retrieves the service provider QoS score as reputation that is stored into QoS Database.

5.4 Communication between SLAAgent Components

The communication between the components of the SLAAgent framework is based on SOA based architecture which is a channel of communication between the main pillars of SOA i.e. service consumers, service providers and service registries. Within the actual structure of SOA paradigm consumers normally find the service providers and service details from services registries and then communicate directly with service providers and receive the services from them, this role is performed by SLAAgent in the proposed framework.

Due to a number of services required to the consumers at the same time, it becomes difficult for them to search different service providers and service details. It is also a tiring task to communicate with service providers manually in order to fulfill their composite requirements. Therefore, an intermediate component called SLAAgent is

proposed within the SOA architecture which helps to automate the search, selection and communication process.

In this framework, initially the consumer requirements are provided to the SLAAgent and then the requirements are further divided into relevant services needed. The required services are discovered from the repository of services. Based on the available services in the services registry, the list of different composite service plans are generated, then the composite plans are offered to the consumer, while the consumer has two options either to select a particular composite service plan out of the auto generated plans or may request any change in the requirement. Once the consumer agrees on the selected plan then the consumer can buy the service or composite set of services and finally the consumer can contribute the service usage experience with the help of monitoring tools to the SLAAgent for building the reputation score of service providers.

5.4.1 Use of Concept Elements in Framework Components

The basic concept elements (defined in Chapter 3) are used in the framework components are specified in Table 5-1.

Table 5-1 : Basic Concept Elements used in SLAAgent

Basic Concept Element	Set/Tuple Representation
Agreement Name	*AN = {"Agreement Name"}*
Agreement Template	$ATEMP = \{ATEMP_1, ATEMP_2, \ldots ATEMP_{atempn}\}$
Agreement Terms	$ATER = \{ATER_1, ATER_2, \ldots ATER_{atern}\}$
Purpose	SLAPU ={"Purpose of SLA"}
Validity Period	*SLAVP= (SD, ED)*
SLA Start Date	*SLASD= (T, D, M, Y)*
SLA End Date	*SLAED= (T, D, M, Y)*
Service Provider	$SP = \{SP_1, SP_2, \ldots SP_{spn}\}$
Service Consumer	$SC = \{SC_1, SC_2, \ldots, SC_{scn}\}$
Third Party	$TP = \{TP_1, TP_2, \ldots TP_{tpn}\}$
Signatory Party	$SIP = \{SIP_1, SIP_2, \ldots SIP_{sipn}\}$

Supporting Party	$SUP= \{SUP_1, SUP_2, \ldots SUP_{supn}\}$
Agreement Initiator	$AIN= \{AIN_1, AIN_2, \ldots AIN_{ainn}\}$
Agreement Responder	$ARES= \{ ARES_1, ARES_2, \ldots ARES_{aresn} \}$
Service Description Terms	$SDT= (S, I, SOP, O)$
Service	$S= \{S_1, S_2, \ldots S_{sn}\}$
Inputs to Service	$I = \{I_1, I_2, \ldots, I_{in}\}$
Outputs of Service	$O = \{O_1, O_2, \ldots, O_{on}\}$
Service Operations	$SOP = \{SOP_1, SOP_2, \ldots\ldots, SOP_{sopn}\}$
Composite Service	$CS= \{CS_1, CS_2, \ldots, CS_{csn}\}$
Service Properties	$SPRO= \{SPRO_1, SPRO_2 , \ldots , SPRO_{spron}\}$
SLA Parameter	$SLAPAR= \{SLAPAR_1, SLAPAR_2 , \ldots , SLAPAR_{slaparn}\}$
Metric	$MET= \{ MET_1, MET_2 , \ldots , MET_{metn}\}$
Measurement Directive	$MD= \{MD_1, MD_2 , \ldots , MD_{mdn}\}$
Function	$FUN= \{FUN_1, FUN_2 , \ldots , FUN_{funn}\}$
Any Attribute	$AA= \{AA_1, AA_2 , \ldots , AA_{aan}\}$
Obligations	$OB= \{OB_1, OB_2 , \ldots , OB_{obn}\}$
Service Scope	$SCOP= \{SCOP_1, SCOP_2 , \ldots , SCOP_{scopn}\}$, $Sub\_SCOP= \{Sub\_SCOP_1, Sub\_SCOP_2, \ldots, Sub\_SCOP_{sub\_scopn}\}$
Service Level Objectives	$SLO= \{SLO_1, SLO_2, \ldots , SLO_{slo}\}$, $Sub\_SLO= \{Sub\_SLO_1, Sub\_SLO_2, \ldots , Sub\_SLO_{sub\_slon}\}$
Action Guarantee	$AG= \{AG_1, AG_2, \ldots , AG_{ago}\}$ $Sub\_AG= \{Sub\_AG_1, Sub\_AG_2, \ldots , Sub\_AG_{sub\_gn}\}$
Penalties	$PEN= \{PEN_1, PEN_2 , \ldots , PEN_{penn}\}$, $Sub\_PEN= \{Sub\_PEN_1, Sub\_PEN_2 , \ldots , Sub\_PEN_{sub\_penn}\}$
Optional Service	$OS= \{OS_1, OS_2 , \ldots , OS_{osn}\}$, $Sub\_OS= \{Sub\_OS_1, Sub\_OS_2 , \ldots , Sub\_OS_{sub\_osn}\}$

Restrictions	$RES = \{ RES_1, RES_2 , \ldots , RES_{resn}\}$, $Sub\_RES = \{Sub\_RES_1, Sub\_RES_2 , \ldots , Sub\_RES_{sub\_resn}\}$
Exclusions	$EXC = \{ EXC_1, EXC_2 , \ldots , EXC_{excn}\}$, $Sub\_EXC = \{Sub\_EXC_1, Sub\_EXC_2 , \ldots , Sub\_EXC_{sub\_excn}\}$
QoS Terms	$QoS\_TERM = \{ QoS\_TERM_1, QoS\_TERM_2 , \ldots , QoS\_TERM_{qos\_termn}\}$
QoS Score	$QoS\_SCORE = \{ QoS\_SCORE_1, QoS\_ SCORE_2 , \ldots , QoS\_ SCORE_{qos\_scoren}\}$
Requirement	$REQ = \{REQ_1, REQ_2, \ldots, REQ_{reqn}\}$ $Sub\_REQ = \{Sub\_REQ_1, Sub\_REQ_2, \ldots , Sub\_REQ_{sub\_reqn}\}$

5.4.2 Inputs/Outputs and Operations of SLAAgent Components

The inputs, operations and outputs of the SLAAgent components are explained in the Table 5-2 to Table 5-9.

Table 5-2 : Inputs/Outputs and Operations for UDDI

COMPONENT: UDDI		
Sub-Component: UDDI's own API		
Description: Stores the service descriptions given by service providers about the services		
Input	Operation	Output
SDT	Store_Service_Descriptions( )	Notification
Comments: UDDI stores service descriptions into Services Registry		

Table 5-3 : Input/Output and Operations for Service Provider

COMPONENT: Service Provider (SP)		
Sub-Component: Service Provider's own API		
Description: Publishes service descriptions to UDDI		
Input	Operation	Output
SDT	Publish_Service_Descriptions()	Notification
Comments: Service descriptions published into UDDI by service provider		

Table 5-4 : Input/Output Operations for Controller

COMPONENT: Controller (C)		
Sub-Component: Send (C:1)		
Description: Sends request to SP, SC, UDDI and any framework component		
Input	Operation	Output
SC,SP,UDDI, RP, SM, SLAM, QoSM, QoSD	Send ( )	SC,SP,UDDI, RP, SM, SLAM, QoSM, QoSD
Comments: Any request is sent to SP, SC, UDDI and any framework component		
Sub-Component: Receive (C:2)		
Description: Receives request from SP, SC, UDDI and any framework component		
Input	Operation	Output
SC,SP,UDDI, RP, SM, SLAM, QoSM, QoSD	Receive( )	SC,SP,UDDI, RP, SM, SLAM, QoSM, QoSD
Comments: Any request is received from SP, SC, UDDI and any framework component		

Table 5-5 : Input/Output and Operations for Requirement Processor

COMPONENT: Requirement Processor (RP)		
Sub-Component: Discover Service Provider (RP:1)		
Description: Discovers Service Provider from UDDI		
Input	Operation	Output
Si , I (key-value)	Discover_Service_Provider( )	SP_{spn} , S_{sn}

Comments: Retrieves list of similar services from UDDI		
Sub-Component: Discover Service Provider QoS (RP:2)		
Description: Discovers QoS information from service provider QoS Database		
Input	Operation	Output
S_i , SP_i	Discover_Service_Provider_QoS( )	S_{sn}, SP_{spn}, QoS
Comments: Retrieves QoS information of service provider from QoS Database for particular service.		
Sub-Component: Discover Composite Services with QoS (RP:3)		
Description: Discovers Composite Services with QoS		
Input	Operation	Output
S_{sn}	Discover_Composite_Service_with_QoS( )	CS_{csn}, SP_{spn}, QoS
Comments: Retrieves composite services plan with QoS information.		
Sub-Component: Request Service Level Agreement (RP:4)		
Description: Requests Service Provider for Service Level Agreement		
Input	Operation	Output
S_i , SP_i	Request_Service_Level_Agreement( )	SLA
Comments: Returns the Service Level Agreement for particular service from service provider.		
Sub-Component: Requirement Preferences (RP:5)		
Description: Service consumer gives requirement preferences for required composite services plans		
Input	Operation	Output
REQ_i , S_i	Requirement_Preferences( )	Formatted REQ_i
Comments: Returns the formatted requirement preferences for filtered composite services plan		

Table 5-6 : Input/Output and Operations for Service Manager

COMPONENT: Service Manager (SM)
Sub-Component: Service Development/Preparation (SM:1)
Description: Customizes service development/preparation according to service consumer requirements

Input	Operation	Output
S_i , I (key-value)	Service_Development_Preparation()	S_i
Comments: Custom configured service descriptions returned from service provider		
Sub-Component: Service Composition (SM:2)		
Description: Creates composite service plan based more than one service Required		
Input	Operation	Output
S_1,..S_{sn}	Service_Composition()	CS_{csn}
Comments: Creates composite service plan based one more than one services		
Sub-Component: Composition Filter (SM:3)		
Description: Filters the composite services plans according to service consumer Preferences.		
Input	Operation	Output
CS_{csn} , REQ	Composition_ Filter()	CS_i
Comments: Returns filtered composite services plans according to service consumer preferences.		
Sub-Component: Execution (SM:4)		
Description: Executes Services if Consumer Agreed to receive the Services		
Input	Operation	Output
Si, I (key-value), SLA	Execution()	Notification
Comments: Execution of service starts and SLA contracts start and monitoring is started		
Sub-Component: Decommission (SM:5)		
Description: Stops services safely during decommission		
Input	Operation	Output
S_i	Decommission()	Notification
Comments: Stops service functionality safely.		

Table 5-7 : Input/Output and Operations for SLA Manager

COMPONENT: SLA Manager (SLAM)		
Sub-Component: Define SLA Template (SLAM:1)		
Description: Defines SLA Template		
Input	Operation	Output
SP_{spn} , S_i	Define_SLA_Template()	SLA
Comments: Service provider defines and provides SLA for the requested service to SLAAgent		
Sub-Component: SLA Negotiation (SLAM:2)		
Description: Initiates negotiation between service consumer and service provider		
Input	Operation	Output
S_i , SLA, Service Provider SP, Service Consumer SC	SLA_Negotiation ( )	SLA
Comments: SLA for specific service being negotiated, and finally agreed mutually between service provider and service consumer.		
Sub-Component: SLA Establishment (SLAM:3)		
Description: Establishes SLA for the services		
Input	Operation	Output
S_i , SLA	SLA_Establishment ( )	Notification
Comments: SLA established		
Sub-Component: SLA Deployment (SLAM:4)		
Description: Deploys SLAs		
Input	Operation	Output
S_i , SLA	SLA_Deployment( )	Notification
Comments: SLA deployed		
Sub-Component: SLA Termination (SLAM:5)		
Description: Terminates SLAs		
Input	Operation	Output
S_i , SLA	SLA_Termination( )	Notification
Comments: SLA terminated		
Sub-Component: Enforce Penalties for SLA Violation (SLAM:6)		
Description: Enforces penalties for SLA violation		

Input	Operation	Output
S_i , SLA, SP, SC	Enforce_Penalties_for_SLA_Violation()	Notification
Comments: Penalties for SLA violation enforced.		

Table 5-8 : Input/Output and Operations for QoS Monitor

COMPONENT: QoS Monitor (QoSM)		
Sub-Component: Service Level Measurement (QoSM:1)		
Description: Performs Service Level Measurements		
Input	Operation	Output
S_i , SLA, SP, SC	Service_Level_Measurement( )	Notification
Comments: Service Level Measurement remains in action		
Sub-Component: Monitor SLA Violation (QoSM:2)		
Description: Monitors SLAs Violations		
Input	Operation	Output
S_i , SLA, SP, SC	Monitor_SLA_Violation( )	Notification
Comments: SLA violation monitored		
Sub-Component: Feedback Receiver (QoSM:3)		
Description: Receives Feedback from monitoring tools		
Input	Operation	Output
QoS_Terms , SP_i , S_i	Feedback_Receiver()	Notification
Comments: Feedback in the form of Rating Score (0 to 10) received from monitoring tools about the service usage		
Sub-Component: Reputation Builder (QoSM:4)		
Description: Builds reputation of service providers		
Input	Operation	Output
QoS_Terms, Si, SP	Reputation_Builder( )	Notification
Comments: Reputation score in the form of QoS score calculated for service provider		

Table 5-9 : Input/Output and Operations for QoS Database

COMPONENT: QoS Database (QoSD)		
Sub-Component: Store (QoSD:1)		
Description: Stores information about service provider reputation(QoS score)		
Input	Operation	Output
QoS, Si, SP	Store( )	Notification
Comments: Reputation (QoS) of service provider stored into Database.		
Sub-Component: Retrieve (QoSD:2)		
Description: Retrieves reputation(QoS score) of service provider		
Input	Operation	Output
Si, SP	Retrieve( )	Notification
Comments: Reputation (QoS score) of service provider retrieved.		

5.4.3 Dependency between Framework Component Elements

The dependency of the SLAAgent components is defined in Table 5-10, which shows the possible connections between sub-components.

Table 5-10 : SLAAgent Components Dependency

Component/Element	Dependent On
RP:1	UDDI
RP:2	UDDI, QoSD:2
RP:3	QoSD:2, SM:2
RP:4	SLAM:1
RP:5	RP:3
SM:1	UDDI, SP
SM:2	UDDI, QoSD:2
SM:3	SM:2, RP:5
SM:4	SP, SC, SM:3, SLAM:3, SLAM:4:
SM:5	SP, SC, SLAM:5
SLAM:1	SC, RP:4, SP
SLAM:2	SC, RP:4, SLAM:1, SP
SLAM:3	SM:4
SLAM:4	SM:4
SLAM:5	SM:5, SLAM:6
SLAM:6	SLAM:5
QoSM:1	SM:4, SLAM:1
QoSM:2	SLAM:5, SLAM:1
QoSM:3	SC, SLAM:1, SP
QoSM:4	SC, QoSM:1, QoSM:2, QoSM:3, SP
QoSD:1	QoSM:4
QoSD:2	QoSD:1

5.5 Components Operation Algorithms

The operation Algorithms for each SLAAgent component are defined in the below sections.

5.5.1 Component Algorithms for UDDI:

The UDDI component is an external component, it is independently implemented by venders of the UDDI. This framework component stores the service descriptions given by service providers about the services. Its Algorithm is given in Table 5-11.

Table 5-11 : Algorithm for Store Service Descriptions

Algorithm 01: Store Service Descriptions
Abstract Detail: **Description**: UDDI stores service descriptions provided by service providers **Input:** A. Service Name and Descriptions **Output**: UDDI stores service Names and Descriptions **Concrete Steps:** **Algorithm Begin** **Input:** SDT(S_i ,I, OP) **Operation:** *Store_Service_Descriptions()* **Statements Begin** // UDDI's own Implementation steps here //UDDI Specification API for service publication of a given WSDL **Statements End** **Output:** Service Name and Descriptions (WSDL) Published on UDDI. using UDDI API **Algorithm End**

5.5.2 Component Algorithms for Service Provider

The service provider (SP) is the provider of services, it is an external component and provides independent implementation of Services and way of communication with UDDI, Service Consumers and SLAAgent. Its Algorithm is defined in Table 5-12.

Table 5-12 : Algorithm for Publish Service Descriptions to UDDI

Algorithm 02: Publish Service Descriptions to UDDI
Abstract Detail: **Description**: Service providers submit the Names and Description of service to UDDI Registry **Input:** A. Name and Description of Service **Output**: Service Description sent to UDDI by Service Provider **Concrete Steps:** **Algorithm Begin** **Input:** SDT (S_i , I, OP) **Operation:** *Publish_Service_Descriptions()* **Statements Begin** //Service Providers Publish Service Description (WSDL) using their own Web //Service Framework Implementation **Statements End** **Output:** Service Name and Description (WSDL) sent to UDDI using client side API by Service Providers **Algorithm End**

5.5.3 Component Algorithms for Service Consumer

The Interaction of service consumer with UDDI and service provider is carried out on the behalf of SLAAgent, hence the algorithms related with Requirement Processor are invoked by service consumer with the help of Controller component of SLAAgent.

5.5.4 Component Algorithms for Controller

All SLAAgent framework components communicate to each other via Controller (C). The Controller Algorithms of sending and receiving requests are defined in Table 5-13 and Table 5-14 respectively.

Table 5-13 : Algorithm for Send

Algorithm 03: Send (C:1)
Abstract Detail: **Description**: Controller sends any request message to any framework component, service provider, service consumer or UDDI. **Input:** A. Message to be sent to Framework Components B. Message to be Sent to Service Provider, Service Consumer, UDDI **Output**: Request Message sent to Service Provider, Service Consumer, UDDI or Framework Components. **Concrete Steps:** **Algorithm Begin** **Input:** I **Operation:** *Send (SP,SC, UDDI, RP, SM, SLAM, QoSM, QoSD, I)* **Statements Begin** //Implementation steps to follow to send message to: (SP, SC, UDDI, RP, SM //,SLAM, QoSM, QoSD) **Statements End** **Output:** Request message sent to service provider, service consumer and UDDI or framework components. **Algorithm End**

Table 5-14 : Algorithm for Receive

Algorithm 04: Receive (C:2)

Abstract Detail:

Description: Controller receives any request message from any framework component, service provider, service consumer or UDDI.

Input:

A. Message to be received from SLAAgent Components

B. Message to be received from Service Provider, Service Consumer, UDDI

Output: Request Message received from Service Provider, Service Consumer, UDDI or SLAAgent Components.

Concrete Steps:

Algorithm Begin

Input: I

Operation:

Receive (SP, SC, UDDI, RP, SM, SLAM, QoSM, QoSD, I)

Statements Begin

// Implementation steps to follow to receive message from : (SP, SC, UDDI,

// RP, SM, SLAM, QoSM, QoSD)

Statements End

Output:

Request Message received from Service Provider, Service Consumer, UDDI or SLAAgent Components.

Algorithm End

5.5.5 Component Algorithms for Requirement Processor

Requirement Processor (RP) receives and refines service consumer requests according to services categories required. Requirement Processor discovers service providers and service Descriptions from UDDI, discovers service provider QoS information from Reputation Database, discovers composite services with QoS, Requests Service Level Agreements for services and defines the Requirement Preferences. Algorithms for Requirement Processor are defined in Table 5-15, Table 5-16, Table 5-17, Table 5-18 and Table 5-19.

Table 5-15 : Algorithm for Discover Service Provider

Algorithm 05: Discover Service Provider (RP:1)
Abstract Detail: **Description**: Requirement Processer searches service provider and service descriptions from UDDI for service consumer **Input:** A. Name of Service B. Inputs for Service **Output**: Requirement Processor retrieves list of similar services from UDDI for a required type of service.
Concrete Steps:
Algorithm Begin **Input:** S_i , I ***Operation:*** *Discover_Service_Provider (Si , I)* **Statements Begin** // Programming Implementation steps for: Requirement Processor to search // the Required Service from UDDI **Statements End** **Output:** Requirement Processor retrieves list of similar Services from UDDI for a required type of Service.
Algorithm End

Table 5-16 : Algorithm for Discover Service Provider QoS

Algorithm 06: Discover Service Provider QoS (RP:2)

Abstract Detail:

Description: Requirement Processer searches QoS information from service provider reputation Database

Input:

A. Name of Services

B. Name of Service Provider

Output: Requirement Processor retrieves QoS information of service provider from reputation Database for particular service.

Concrete Steps:

Algorithm Begin

Input: S_i , SP

Operation:

Discover_Service_Provider_QoS (S_i , SP_i)

Statements Begin

1. Implementation steps for retrieve QoS information from reputation Database.
2. Implementation steps for: Forward QoS to service consumer

Statements End

Output:

Requirement Processor retrieves QoS information of service provider from Reputation Database for particular service.

Algorithm End

Table 5-17 : Algorithm for Discover Composite Services with QoS

Algorithm 07: Discover Composite Services with QoS (RP:3)

Abstract Detail:

Description: Service consumer requests to SLAAgent for composite service description with QoS score

Input: A. Name of Services

B. Inputs for required number of Services

Output: Response from SLAAgent Returns list of Composite Services for requested Service plans along with QoS information

Concrete Steps:

Algorithm Begin

Input: S_i, .., S_{sn} , S_i , I

Operation:

Discover_Composite_Service_with_QoS (S_i, .., S_{sn} , S_i , I)

Statements Begin

1. Service Consumer Sends Request to Controller by sending (S_i,...S_{sn}, I)
2. Controller forwards the Request to Requirement Processor
3. Requirement Processor Searches Required Services from UDDI one by one
4. For each category of Services found, the Requirement Processor finds QoS of each Service Provider Category from Service Provider Reputation Database.
5. Finally, Requirement Processor forwards, services and QoS information to service composition element of Service Manager that returns the List of Composite services plan along with corresponding QoS information for each service provider.

Statements End

Output:

Response from SLAAgent Returns List Composite Services plans for more than one Service Required with QoS Information in a plan

Algorithm End

Table 5-18 : Algorithm for Request Service Level Agreement

Algorithm 08: Request Service Level Agreement (RP:4)

Abstract Detail:

Description: Service consumer requests Service Level Agreement for particular service from service provider

Input:

A. Name of Services

B. SLA requested for I (Input Requirement)

Output: Response from SLAAgent returns SLA for requested Service

Concrete Steps:

Algorithm Begin

Input: S_i , I

Operation:

Request_Service_Level_Agreement (S_i , I)

Statements Begin

// Request passed from Requirement Processor to Service Provider

Statements End

Output:

Response from SLAAgent returns Service Level Agreement for particular service

Algorithm End

Table 5-19 : Algorithm for Requirement Preferences

Algorithm 09: Requirement Preferences (RP:5)
Abstract Detail: **Description**: Service Consumer provides the Requirement Preferences for the required Services **Input:** A. Name of Services B. Preferred Service Requirements **Output**: Response from SLAAgent Returns formatted Requirement Preferences for Composition Filter Component. **Concrete Steps:** **Algorithm Begin** **Input:** S_i , REQ. I **Operation:** *Requirement_Preferences* (S_i , REQ.I) **Statements Begin** // Requirement Preferences passed to Composition Filter (SM:3) for getting // Composition Filter **Statements End** **Output:** Response from SLAAgent Returns formatted Requirement Preferences for Composition Filter Component. **Algorithm End**

5.5.6 Component Algorithms for Service Manager

The Service Manager (SM) deals with services and service providers for making and preparing the services for execution and perform decommission if required. Services Manager Customizes service Development/Preparation according to service consumer requirements, creates composite service plan based more than one service required based on QoS Information, execute services if consumer agreed to receive the services and stops services safely during Decommission. Algorithms for Service Manager are defined in Table 5-20, Table 5-21, Table 5-22, Table 5-23 and Table 5-24.

Table 5-20 : Algorithm for Service Development/Preparation

Algorithm 10: Service Development/Preparation (SM:1)

Abstract Detail:

Description: SLAAgent requests the service provider to reconfigure the service on demand if predefined service is not sufficient to fulfill the requirement of consumer.

Input:

A. Names of Service

B. Inputs to Service

Output: Custom Configured Service Descriptions Returned from Service Provider

Concrete Steps:

Algorithm Begin

Input: S_i , I

Operation:

Service_Development_Preparation (S_i , I)

Statements Begin

// Implementation to send request to Service Provider for Customization of
// Services

Statements End

Output:

Custom configured service descriptions returned from service provider

Algorithm End

Table 5-21 : Algorithm for Service Composition

Algorithm 11: Service Composition (SM:2)

Abstract Detail:

Description: Composes services plans based on QoS information

Input:

A. Name of Services

B. Inputs for required Number of Services

Output: Response from Service Manager component returns list of Composite Services for requested service plans along with QoS information

Concrete Steps:

Algorithm Begin

Input: S_i, .., S_{sn} , S_i.I

Operation:

Service_Composition (S_i , .., S_{sn} , S_i , I)

Statements Begin

1. Service Consumer Sends Request to Controller by sending (S_i,...S_{sn}, I)
2. Controller forwards the Request to Requirement Processor
3. Requirement Processor Searches Required Services from UDDI one by one
4. For each category of Services found, the Requirement Processor finds QoS of each Service Provider Category from Service Provider Reputation Database.
5. Finally, Requirement Processor forwards, Services and QoS information to Service Composition sub-component of Service Manager that returns the List of Composite Services plan along with corresponding QoS information for each Service Provider.

Statements End

Output:

Response from SLAAgent Returns List Composite Services plans for more than one Services Required with QoS Information in a plan

Algorithm End

Table 5-22 : Algorithm for Composition Filter

Algorithm 12: Composition Filter (SM:3)

Abstract Detail:

Description: Composition Filter extracts selected composite service plans from all composite services plans

Input:

A. Composite Services Plans

B. Requirement Preferences

Output: Returns only selected Composite Service plan based on Requirement Preferences of Service Consumer.

Concrete Steps:

Algorithm Begin

Input: S_i , REQ, I, CS_i

Operation:

Composition_Filter (S_i , .., S_{sn} , *REQ, I)*

Statements Begin

1. Sort-out Composite Service plan CS according to Requirement Preference
2. Select only CS plan that matches Requirement Preference

Statements End

Output:

Returns only selected Composite Service plan based on requirement preferences of service consumer.

Algorithm End

Table 5-23 : Algorithm for Execute

Algorithm 13: Execute (SM:4)

Abstract Detail:

Description: SLAAgent executes service/services if consumer satisfied with service Descriptions and SLA Terms, then Monitoring of service and SLA starts after Service starts execution.

Input:

A. Service Inputs

B. Names of Service

C. Operations of Service

D. SLA involved into Service

Output: Execution of Service starts and SLA contracts start and monitoring is started

Concrete Steps:

Algorithm Begin

Input: Si, I, SLA

Operation:

Execute (Si , I, SLA)

Statements Begin

1. If Consumer satisfied with Service Description returned from UDDI, and SLA from Service Provider Then
2. Execute Service with Service Manager Using Service Operations
3. Establish and Deploy SLA
4. Monitor SLA Violation starts

Statements End

Output:

Execution of Service starts and SLA contracts start and monitoring is also started

Algorithm End

Table 5-24 : Algorithm for Decommission

Algorithm 14: Decommission (SM:5)
Abstract Detail: **Description**: Decommission stops functionality of service safely. **Input:** A. Names of Services, S_i **Output**: Stops service functionality safely. **Concrete Steps:** **Algorithm Begin** **Input: S_i** **Operation:** *Decommission (S_i)* **Statements Begin** //Implementation steps to stop Service S_i safely **Statements End** **Output:** Service stopped Safely. **Algorithm End**

5.5.7 Component Algorithms for SLA Manager

SLA manager (SLAM) deals with structure and management of SLAs during and after service provision. Services Manager defines SLA/SLA Templates, initiates Negotiation between service consumer and service provider, establishes SLA for the services, deploys SLAs, terminates SLAs and Enforces Penalties for SLA Violation. The Algorithms of SLA Manager are defined in Table 5-25, Table 5-26, Table 5-27, Table 5-28, Table 5-29 and Table 5-30.

Table 5-25 : Algorithm for Define SLA Template

Algorithm 15: Define SLA Template (SLAM:1)

Abstract Detail:

Description: Service provider defines SLA for their service to SLAAgent

Input:

A. S_i , SP_i

Output: SLA for specific service given to SLAAgent from Service Providers

Concrete Steps:

Algorithm Begin

Input: S_i

Operation:

Define_SLA_Template (SP_i , S_i)

Statements Begin

1. SLA for S_i passed to SLAAgent via Controller
2. The Controller Passes SLA to SLA Manager
3. The SLA Manager reformats SLA using Basic Concept Elements (Chapter3)
4. The SLA Manager Returns SLA to Controller
5. Controller returns SLA to Service Consumer.

Statements End

Output:

SLAAgent receives SLA and Passes to Service Consumer

Algorithm End

Table 5-26 : Algorithm for SLA Negotiation

Algorithm 16: SLA Negotiation (SLAM:2)

Abstract Detail:

Description: Negotiate for offer and counter offer for mutual Agreement between service provider and service consumer

Input:

A. S_i

B. SLA

C. Service Provider SP

D. Service Consumer SC

Output: SLA for specific service being negotiated, and finally agreed mutually between service provider and service consumer.

Concrete Steps:

Algorithm Begin

Input: S_i , SLA, Service Provider SP, Service Consumer SC

Operation:

SLA_Negotiation (S_i , SLA, SP, SC)

Statements Begin

1. Request Service Descriptions from Requirement Processor
2. Request SLA from Service Provider
3. Send Offer/Counter Offer to Service Provider
4. Once Negotiation completed, Execute Service/Services via Service Manager

Statements End

Output:

SLA for specific service being negotiated, and finally agreed mutually between service provider and service consumer.

Algorithm End

Table 5-27 : Algorithm for SLA Establishment

Algorithm 17: SLA Establishment (SLAM:3)

Abstract Detail:

Description: SLAAgent establishes SLA when service consumer agrees to get the service from service provider.

Input:

A. Name of Service

B. Inputs for Service

C. SLA

Output: SLA Established

Concrete Steps:

Algorithm Begin

Input: S_i, I, SLA

Operation:

SLA_Establishment (S_i , I, SLA)

Statements Begin

//Implementation steps of SLA Establishment

Statements End

Output:

SLA established

Algorithm End

Table 5-28 : Algorithm for SLA Deployment

Algorithm 18: SLA Deployment (SLAM:4)

Abstract Detail:

Description: SLAAgent deploys SLA when service consumer agrees to get the service from service provider.

Input:

A. Name of Service

B. Inputs for Service

C. SLA

Output: SLA Deployed

Concrete Steps:

Algorithm Begin

Input: S_i , SLA

Operation:

SLA_Deployment (S_i , SLA)

Statements Begin

// Implementation steps for SLA Deployment

Statements End

Output:

SLA Deployed

Algorithm End

Table 5-29 : Algorithm for SLA Termination

Algorithm 19: SLA Termination (SLAM:5)

Abstract Detail:

Description: Terminates SLA due to either due to SLA validity period completed, or abnormally terminated due to predefined conditions, which terminate the SLA.

Input:

A. Name of Service

B. SLA

Output: SLA Terminated

Concrete Steps:

Algorithm Begin

Input: S_i , SLA Template

Operation:

SLA_Termination (S_i , SLA)

Statements Begin

// Implementation steps to Terminate SLA

Statements End

Output:

SLA Terminated

Algorithm End

Table 5-30 : Algorithm for Enforce Penalties for SLA Violation

Algorithm 20: Enforce Penalties for SLA Violation (SLAM:6)
Abstract Detail: **Description**: Enforces Penalties for SLA Violation **Input:** A. Name of Service B. SLA *C. Service Provider* *D. Service Consumer* **Output**: Penalties for SLA Violation Enforced. **Concrete Steps:** **Algorithm Begin** **Input:** S_i , SLA, SP, SC **Operation:** *Enforce_Penalties_for_SLA_Violation (S_i , SLA, SP, SC)* **Statements Begin** // Implementation steps for Enforcement of Penalties for SLA Violation **Statements End** **Output:** Penalties for SLA Violation enforced. **Algorithm End**

5.5.8 Component Algorithms for QoS Monitor

QoS Monitor (QoSM) is responsible for QoS management and monitoring. QoS Monitor performs Service Level Measurements, monitors SLAs violation, receives Feedback from monitoring Tools and builds reputation of service providers as the QoS score using Fuzzy Inference System defined in Chapter 4. The Algorithms for QoS Monitor component are defined in Table 5-31, Table 5-32, Table 5-33 and Table 5-34.

Table 5-31 : Algorithm for Service Level Measurement

Algorithm 21: Service Level Measurement (QoSM:1)

Abstract Detail:

Description: Performs Service Level Measurement

Input:

A. Name of Service

B. SLA

Output: Service Level Measurement keep checking current system configuration

Concrete Steps:

Algorithm Begin

Input: S_i , SLA, SP, SC

Operation:

Service_Level_Measurement (S_i , SLA, SP, SC)

Statements Begin

// Implementation steps for Service Level Measurement

Statements End

Output:

Service Level Measurement in action of current system configuration check.

Algorithm End

Table 5-32 : Algorithm for Monitor SLA Violation

Algorithm 22: Monitor SLA Violation (QoSM:2)

Abstract Detail:

Description: Monitors SLA Violation

Input:

A. Name of Service

B. SLA

C. Service Provider

D. Service Consumer

Output: SLA Violation Monitored

Concrete Steps:

Algorithm Begin

Input: S_i , SLA, SP, SC

Operation:

Monitor_SLA_Violation (S_i , SLA, SP, SC)

Statements Begin

1. Get SLOs from SLA
2. Compare SLOs QoS Metric
3. Determine SLOs satisfies the QoS Metric

Statements End

Output:

SLA Violation Monitored

Algorithm End

Table 5-33 : Algorithm for Feedback Receiver

Algorithm 23: Feedback Receiver (QoSM:3)

Abstract Detail:

Description: Receive Feedback from Service Consumer about the Service used

Input:

A. List of QoS Terms

B. Name of Service

C. Service Provider

Output: Feedback in the form of Rating Score (0 to 10) Received from Service Consumer about the Service Provider Reputation

Concrete Steps:

Algorithm Begin

Input: QoS_Terms, SP_i , S_i

Operation:

Feedback_Receiver (QoS_Terms, SP_i , S_i)

Statements Begin

// Implementation of integrating the monitoring tools for receiving the

// individual QoS Term metrics score for calculating the QoS based on Fuzzy

// Inference System in Reputation Builder sub-component (QoSM:4)

Statements End

Output:

Feedback in the form of Rating Score (0 to 10) received from monitoring tools about the individual QoS Term metrics

Algorithm End

Table 5-34 : Algorithm for Reputation Builder

Algorithm 24: Reputation Builder (QoSM:4)

Abstract Detail:

Description: Builds Reputation as QoS Score of Service Provider by using Fuzzy Inference System Steps.

Input: A. List of QoS Terms

B. Name of Service

C. Service Provider

D. Fuzzy Inference System Steps

Output: Reputation Score Calculated for Service Provider

Concrete Steps:

Algorithm Begin

Input: QoS_TERM_i , S_i , SP

Operation:

Reputation_Builder ( )

Statements Begin

1. Determine the Number of Inputs and Outputs
2. Define Input Membership Functions
3. Define Output Membership Functions
4. Determine the Number of Fuzzy rules
5. Fuzzify inputs
6. Combining the Fuzzified inputs
7. Compute the rule strength
8. Define Fuzzy Associative Memory
9. Find Consequence of the Rule
10. Aggregate Rule Outputs
11. Defuzzify output

// Mockup Implementation is given in Appendix-B, Table 10-1

Statements End

Output:

Reputation Score Calculated for Service Provider

Algorithm End

5.5.9 Component Algorithms for QoS Database

QoS Database stores the QoS ranking Information for different service providers in the form of service provider reputation. The Algorithms for storing the service provider reputation and retrieving the service provider reputation are shown Table 5-35 in and Table 5-36 respectively.

Table 5-35 : Algorithm for Store

Algorithm 25: Store (QoSD:1)
Abstract Detail: **Description**: SLAAgent stores service provider reputation into QoS Database **Input:** A. Names of Service B. Name of Service Provider C. QoS information of Service Provider in form of QoS Score **Output**: QoS score of Service Provider stored into QoS database
Concrete Steps: **Algorithm Begin** **Input:** S_i , SP_i , QoS_Score **Operation:** *Store (S_i , SP_i , QoS_Score)* **Statements Begin** QoS_Table.S_i= S_i , QoS_Table.SPi=SP_i QoS_Table.QoS_Score=QoS_Score **Statements End** **Output:** QoS Score of Service Provider stored into QoS database **Algorithm End**

Table 5-36 : Algorithm for Retrieve

Algorithm 26: Retrieve (QoSD:2)
Abstract Detail: **Description**: SLAAgent retrieves service provider QoS Score as reputation from QoS Database **Input:** A. Names of Service B. Name of Service Provider **Output**: QoS Score of service provider retrieved from QoS database **Concrete Steps:** **Algorithm Begin** **Input:** S_i , SP_i **Operation:** *Retrieve (S_i , SP_i)* **Statements Begin** // SQL/Database Query for retrieval of QoS Score from particular database table column **Statements End** **Output:** QoS score retrieved from QoS database **Algorithm End**

5.6 Steps for using the SLAAgent Framework

For using the SLAAgent framework, following steps should be followed:

Step 1: Requirement Submission

A service consumer needs to provide the names and number of services required. The service consumer is given a list of services from SLAAgent for selection. The requirement process performs discovery of service provider either for one or more number of services using (RP:1). The discovery of service provider with QoS information will be processed by (RP:2). The request for composite services plan with

QoS information will be processed by (RP:3). The request for Service Level Agreement of each service for which the service consumer is interested will be processed by (RP:4). The service consumer requires to define the Requirement Preferences (RP:5) for the required filtered outputs will be performed by (SM:3) from the composite services plans given from (SM:2).

Step 2: Output of Requirement Process

Based on the service consumer requirements, which can involve one or more number of services being requested along with Requirement Preferences (RP:5), the discovery of each single service required from the UDDI will be performed using (RP:1) and the output will be the list of different service providers for the same required service. The discovery of service provider with QoS information will be performed using (RP:2, QoSD:2) and the output will be list of service providers for a service with QoS information. For two or more services required to the service consumer in the composite services plan along with QoS information the sub-components used will be (RP:3, SM:2, and QoSD:2) and the output will be a list of all possible composite service plans along with overall QoS of the composite service plan. The process of Requirement Preferences will be started using (RP:5) and (SM:3) and the output will be the filtered composite services plans from the list of all composite services plans.

Step 3: Request of Service Level Agreement

Every service provider of each service describes the terms and conditions of the service being offered in the form of Service Level Agreement (SLAs). A service consumer can request a Service Level Agreement for each service he wants to use. The request for Service Level Agreement will be performed using (RP:4, SLAM:1) and the output will be a Service Level Agreement of the service that will be further re-formatted by SLAAgent using basic concept elements from Chapter 3 by sub-component (SLAM:1).

Step 4: Negotiation

Once the service consumer required service or services are discovered (RP:1), with QoS information (RP:2) within a composite services plan (RP:3, SM:2) having corresponding Service Level Agreement (RP:4, SLAM1), it is possible that the service

consumer may slightly change the requirements and may want to negotiate for offers and counter offers for this service provider, this negotiation will be performed using (SLAM:2) and the output will settle the negotiation of the offer and counter offer between service provider and service consumer. While based on the changes in the requirement due to negotiation, the service can also be altered or re-prepared on custom needs (SM:1).

Step 5: Execution

Once all the services and SLAs are acceptable to service consumer (SM:3), then SLAAgent can trigger the execution of services using (SM:4) followed by SLA Establishment (SLAM:3) and SLA Deployment (SLAM:4).

Step 6: Termination

The execution of the service or services plan can be terminated on due time that is explicitly defined in SLAs (SLAM:5) and then services can be stopped safely as Decommission of services (SM:4). If the services are not provided adequately according to SLA conditions or terminated abnormally certain Enforcement of Penalties for SLA violations will be performed using (SLAM:6).

Step 7: QoS Monitoring

The monitoring of services starts as the services start execution, QoS Monitoring maintains the record of configuration of services defined in SLAs and keeps checking those configurations at runtime (QoSM:1). If any violation of SLA is found during the service execution or after the termination of service, then SLA Violation is Monitored, (QoSM:2) accordingly. After the completion of services used, the feedback is received from the service consumer about the experience of the services used (QoSM:3) using QoS monitoring tools. Finally, the overall reputation of service provider as QoS score about their offered services is calculated (QoSM:4) using Fuzzy Inference Method from Chapter 4, and stored into QoS Database permanently (QoSD:1).

5.7 Chapter Summary

This chapter defined the SLAAgent framework as an extension of SOA and illustrated the High Level and Low Level diagrams of the framework and defined the communication between SLAAgent components. The SLAAgent framework is designed to use the refined SLA elements from Chapter 2 (Table 2-6), SLA lifecycle stages from (Table 2-20), QoS terms from (Table 2-43) and Fuzzy Inference System process steps for QoS calculation from Chapter 4 (Section 4.4). The algorithms for each component and sub-components of SLAAgent framework, operations and the necessary steps for using the SLAAgent framework are discussed in detail in this chapter.

6. Use Case Example

6.1 Introduction

There are a number of service providers offering their services that are available on the Internet for many domains of interests. However, due to the heterogeneous nature of their structures and technological differences, the selection on the basis of their QoS for individual and composite services becomes a difficult challenge. In order to reduce the efforts of processing steps needed in composition of multiple services on the basis of QoS, it is necessary to formalize and manage the services, SLAs and Quality of Service terms adequately. The proper formalization and management of services, SLA Structures and QoS mechanism can solve the problems related with selection, monitoring and composition of services efficiently and effectively.

This chapter works through a use case example, which shows how the SLAAgent framework proposed in this thesis can be used for the use case scenario requiring one or more services in a composite services plan.

6.2 Computational Service Use Case Scenario

The use case scenario used in this chapter is based on computational services which may use a wide variety of Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). In the use case scenario, a consumer may require to search and select one or more best possible set of services individually for their computational needs. Instead of searching all the required services separately, the consumer may want to get the group of services through a services portal which should produce a complete, optimized and well scheduled composite services plan. As the consumer agrees on the automatically generated proposed composite services plan, then all the inclusive services of a plan should be obtained automatically as well as simultaneously, so that consumer may not face any inconvenience during the concurrent use of all these service.

The satisfied acceptance of a service or a set of services used by the consumer from the same or different services providers depends on the Quality of Service, that should be

monitored and evaluated explicitly with the help of Service Level Agreements defined for the required services. The assessment of the services and their SLAs should be based on the monitoring assessment results of the services performed by either service providers or the third party monitoring tools on the regular basis. The overall satisfaction with the Quality of Service and service provider reputation of an existing consumer can also help the new consumers to select the same services with confidence and peace of mind.

In the use case scenario from Cloud computing, a consumer can be an individual person who requires the computing services for personal use, or a businessperson in a company. A company may require more expensive and reliable services because of their needs, while an ordinary consumer under a limited budget may require some cheaper and reliable services.

The use of the SLAAgent proposed in this study facilitates all types of consumers who want to select and buy the computing services according to their budget and preferences.

Scenario: A new private company wants to buy computing services which includes a Web Hosting Server and a File Storage Server that lie within the range of company's fixed budget for a particular period of time. The required services can be bought from the same or different service providers.

Example: Mr. A.S, an IT manager of a company wants to buy the services of (1) Web Hosting Server for company website, and (2) File Storage Server for the company's clients data. The services should start from the dates starting 30 March 2016 to 30 March 2017. The company has a fixed budget of £1200 for the required computing services.

Using the proposed SLAAgent framework of this study, the IT Manager wants to get the list of all Web Hosting Server providers with their prices and technical features, the list of all File Storage Server providers along with prices and technical features. He also

wants to know the possible combination of the services the company requires along with the information about the total cost of each combination of services, Service Level Agreements for the services, and their corresponding Quality of Service in the form of service provider reputation score based on two QoS terms defined within the SLA and one QoS term defined without SLA. The IT Manager prefers the Quality of Service for File Storage Server to be more reputable and reliable than the Web Hosting Server because he prefers the company's File Storage Server data to be more safe and efficient than the Web Hosting Server for the company website. The IT Manager wants to view the service provider's QoS score based on three QoS terms separately for each service as well as the combination of QoS score for the whole services plan. Once he discovers the required services in a composite service plans, he needs to select a particular service plan, then he wants to buy the services from the selected composite service plan. Finally, on the basis of usage of services according to service commitments defined in the SLAs, he wants to contribute to disclose the usage experience of the services with the help of monitoring tools integrated in the SLAAgent framework.

6.2.1 Use of SLAAgent in Computational Service Scenario

The list of services (shown in Table 6-1) and service providers (shown in Table 6-2) used in the use case example are denoted using basic concept elements from Chapter 3 for short and easy names throughout the demonstration of use case example in this chapter.

For each service that is provided from different service providers there is a QoS score associated with each service, which is required for the selection of service in a composite service plan. It is assumed that in this use case example each service required to the consumer has QoS score that is calculated on the basis of Fuzzy Inference System that was discussed in Chapter 4, and the QoS calculations based on Fuzzy Inference System are based on any three QoS terms metrics from Table 2-43. The samples of QoS scores used in the use case example are taken from mockup calculations shown in Appendix-B, Table 10-3.

Table 6-1: Use of Basic Concept Elements for Services

No.	Name of Service	Basic Concept Element (Service: S)
1	Web Hosting Server	S_1
2	File Storage Server	S_2
3	Email Server	S_3
4	Database Server	S_4
5	SQL Server	S_5

Table 6-2: Use of Basic Concept Elements for Service Providers

No.	Service Provider Name	Basic Concept Element (Service Provider: SP)
1	Amazon	SP_1
2	Google	SP_2
3	Windows Azure	SP_3
4	HP	SP_4

The given use case example from computational services scenario is worked out through the recommended steps of SLAAgent framework and its components (RP, SM, SLAM, QoSM, C and QoSD) from Chapter 5. The follow up of steps (Section 5.6) is given below:

Step 1: Requirement Submission

In this step, Mr. A.S (IT Manager of company) selects the name and number of services he wants. The list of services available in the services Registry that are selected by the IT Manager are shown in Table 6-3. Mr. A.S is interested in two services i.e. Web Hosting Server and File Storage Server. For the required services, Mr. A.S needs to provide the necessary requirements about the services he wants for his company. The Table 6-4 shows the requirement inputs to Web Hosting Server and Table 6-5 shows requirement inputs for File Storage service required to the company. Mr. A.S has a preference over the services to be bought. According to his requirements, he prefers the more reputable and reliable File Storage Server than Web Hosting Server for his

company, so his service requirement preferences under the company's budget are shown in Table 6-6.

Table 6-3: List of Services Available in Services Registry

No.	Service Name	Basic Concept Element	Need this Service	Needs QoS of Service	Needs in Composite Service Plan	Needs SLA
1	Web Hosting Server	S_1	Yes	Yes	Yes	Yes
2	File Storage Server	S_2	Yes	Yes	Yes	Yes
3	Email Server	S_3	No	N/A	N/A	N/A
4	Database Server	S_4	No	N/A	N/A	N/A
5	SQL Server	S_5	No	N/A	N/A	N/A

Table 6-4: Inputs for Web Hosting Server (S_1)

I (Inputs to Service)	Name	Value
I_1	Disk Space	20 GB
I_2	Monthly Bandwidth	100 GB
I_3	Guaranteed Memory	1 GB
I_4	Dedicated IP Addresses	1

Table 6-5: Inputs for File Storage Server (S_2)

I (Inputs to Service)	Name	Example Value
I_1	Maximum Storage Space	10 TB
I_2	Maximum Upload File Size	10 GB
I_3	Files Archived	90 Days
I_4	External Hard Drive Backup	Required
I_5	File Sync	Required

Table 6-6: IT Manager Requirement Preferences

(1st) Service Preference	**(2nd) Service Preference**	**Maximum Budget for all Services**
File Storage Server (S_2)	Web Hosting Server (S_1)	£1200

Step 2: Output of Requirement Process

Based on the requirements provided by Mr. A.S, Table 6-7 shows the detailed QoS score for Web Hosting Service Providers and Table 6-9 shows the total number of Web Hosting Service providers for the requested composite services (RP:1) (Figure 5-3) with QoS Information, (RP:2, QoSD:2) (Figure 5-3), Table 6-8 shows the detailed QoS score for File Storage Service Providers and Table 6-10 shows total number of available File Storage Server Providers required (RP:1) (Figure 5-3) with QoS Information (RP:2, QoSD:2) (Figure 5-3). Table 6-11 shows the total possible combinations of composite services plans for Web Hosting Server and File Storage Server with combined QoS information that Mr. A.S can select and buy (RP:3, SM:2, QoSD:2) (Figure 5-3).

Based on Mr. A.S's requirement preferences, within the available budget of £1200 and preference of File Storage Server efficiency over Web Hosting Server given in Table 6-6 (RP:5) (Figure 5-3), the filtered composite service plans (SM:3) (Figure 5-3) are given in Table 6-12. The filtered composite service plans have a QoS value between (11.24125 to 11.6575) and is within the available budget of £1200. It is sorted on the Quality of Service of File Storage Service providers (S2) in descending order. There are 4 composite service plans filtered, in which the composite service plan 4 has maximum value of QoS score of (7.07875) for File Storage Server provider, which is matching the Mr. A.S requirement preferences.

Table 6-7: Detailed QoS Score for Web Hosting Service Providers

Service Provider Name	**QoS Term-1 (From SLA)**	**QoS Term-2 (From SLA)**	**QoS Term-1 (Without- SLA)**	**QoS Score**
SP_1	7	7	8	7.182813
SP_2	7	6	7	6.141563

SP_3	3	6	9	5.308438
SP_4	10	7	1	4.57875

Table 6-8: Detailed QoS Score for File Storage Service Providers

Service Provider Name	QoS Term-1 (From SLA)	QoS Term-2 (From SLA)	QoS Term-1 (Without- SLA)	QoS Score
SP_1	6	7	9	7.07875
SP_2	6	6	7	6.0375
SP_3	6	5	6	5.204375
SP_4	4	4	10	4.058438

Table 6-9: Algorithm output for Web Hosting Service Providers

Service Provider Name	Disk Space	Monthly Bandwidth	Guaranteed Memory	Dedicated IP Address	Cost	QoS Score
SP_1	50 GB	250 GB	2.5 GB	1	£790	7.182813
SP_2	40 GB	200 GB	2 GB	1	£685	6.141563
SP_3	30 GB	150 GB	1.5 GB	1	£570	5.308438
SP_4	20 GB	100 GB	1 GB	1	£455	4.57875

Table 6-10: Algorithm output for File Storage Service Providers

Service Name	Max. Storage Space	Max. Upload File Size	Files Archived	External Hard Drive Backup	File Sync	Cost	QoS Score
SP_1	25 TB	25 GB	100 Days	Supported	Supported	£700	7.07875
SP_2	20 TB	20 GB	95 Days	Supported	Supported	£600	6.0375
SP_3	15 TB	15 GB	95 Days	Supported	Supported	£500	5.204375
SP_4	10 TB	10 GB	90 Days	Supported	Supported	£400	4.058438

Table 6-11: Composite Service Plan for Web Hosing Server and File Storage Server

Composite Plan	S_1 Providers	S_2 Providers	Total Cost (S_1+ S_2)	S_1 Provider QoS	S_2 Provider QoS	Sum of (S_1 & S_2) Provider QoS
1	SP_1	SP_1	£790+£700 = £1490	7.182813	7.07875	14.26156
2	SP_2	SP_1	£685+£700 = £1385	6.141563	7.07875	13.22031
3	SP_3	SP_1	£570+£700 = £1270	5.308438	7.07875	12.38719
4	SP_4	SP_1	£455+£700 = £1155	4.57875	7.07875	11.6575
5	SP_1	SP_2	£790+£600 = £1390	7.182813	6.0375	13.22031
6	SP_2	SP_2	£685+£600 = £1285	6.141563	6.0375	12.17906
7	SP_3	SP_2	£570+£600 = £1170	5.308438	6.0375	11.34594
8	SP_4	SP_2	£455+£600 = £1055	4.57875	6.0375	10.61625
9	SP_1	SP_3	£790+£500 = £1290	7.182813	5.204375	12.38719
10	SP_2	SP_3	£685+£500 = £1185	6.141563	5.204375	11.34594
11	SP_3	SP_3	£570+£500 = £1070	5.308438	5.204375	10.51281
12	SP4	SP_3	£455+£500 = £955	4.57875	5.204375	9.783125
13	SP_1	SP_4	£790+£400 = £1190	7.182813	4.058438	11.24125
14	SP_2	SP_4	£685+£400 = £1085	6.141563	4.058438	10.2
15	SP_3	SP_4	£570+£400 = £970	5.308438	4.058438	9.366875
16	SP_4	SP_4	£455+£400 = £855	4.57875	4.058438	8.637188

Table 6-12: Algorithm output for Composition Filter

Composite Plan	S_1 Providers	S_2 Providers	Total Cost (S_1+ S_2)	S_1 Provider QoS	S_2 Provider QoS	Sum of(S_1 Provider QoS & S_2 Provider QoS)
4	SP_4	SP_1	£455+£700 = £1155	4.57875	7.07875	11.6575
7	SP_3	SP_2	£570+£600 = £1170	5.308438	6.0375	11.34594
10	SP_2	SP_3	£685+£500 = £1185	6.141563	5.204375	11.34594
13	SP_1	SP_4	£790+£400 = £1190	7.182813	4.058438	11.24125

Step 3: Request of Service Level Agreement

For the most suitable composite service plan matching IT Manager's requirement preferences (i.e. composite service plan 4) the Service Level Agreement of Web

Hosting Service provider (SP_4) and File Storage Service provider (SP_1) are shown in Table 6-13 and Table 6-14 respectively using basic concept element definitions from Chapter 3 and sample SLA data from Table 9-5 for SLAs from HP and Table 9-6 for SLAs from Amazon that are given in Appendix-A.

Table 6-13: SLA from SP4 (Web Hosting Service Provider)

Basic Concept Element	**Basic Concept Element Value**
Agreement Name: AN	AN= {"HP-Web Hosting Server" SLA}
Agreement Template:ATEMP	ATEMP= {"Template No.HPWHS-1"}
Agreement Terms: ATER	*ATER* **={** "Availability: Means degree to which system accessible", "Response Time: Time Required to Complete Request" **}**
Purpose : SLAPU	SLAPU ={"Web Hosting Server Contract"}
Service Provider: SP	SP= {"HP Cloud Services"}
Service Consumer: SC	SC= {"Mr. A.S"}
Third Parties: TP	TP={"Payment Processor: MasterCard"}
Signatory Party: SIP	SIP ={"Service Provider: HP Cloud Services", "Service Consumer: Mr. A.S"}
Supporting Party: SUP	SUP={"Third Party: Payment Processor"}
Agreement Initiator: AIN	AIN={"HP Cloud Services"}
Agreement Responder: ARES	ARES={"Mr. A.S"}
Validity Period: VP	VP= (SD, ED)
Starting Date: SD	SD= ("10:00", "30"," March", "2016")
Ending Date: ED	ED= ("10:00", "30"," March", "2017")
Service: S	S={"Web Hosting Server"}
Inputs-to-Service: I	I={"Disk Space: 20 GB", "Monthly Bandwidth: 100GB", "Guaranteed Memory: 1 GB", "Dedicated IP Address: 1"}
Outputs-of-Service:O	O={"Disk Space: 20 GB", "Monthly Bandwidth: 100GB", "Guaranteed Memory: 1 GB", "Dedicated IP Address: 1", "Price: £445"}
Service Description Terms:	SDT={"Disk Space", "Monthly Bandwidth",

SDT	”Guaranteed Memory”, “Dedicated IP Address”}
Service Properties: SPRO	SPRO={”Service URL: XYX”, “Port:YYY”,”OS Support: Windows”}
SLA Parameter: SLAPAR	SLAPAR={”Availability”, “Response Time”}
Metric: MET	MET={“Availability: Percentage”, “Response Time: Seconds”}
Measurement Directive:MD	MD={“Availability:1 to 99”, “Response Time:0 to 60”}
Function: FUN	FUN={“Availability-FUN”, ”Response Time-FUN”}
Any Attribute:AA	AA={“OS Support: Windows”}
Obligations: OB	OB={“Service Commitments”, ”Service Credits”, ”Action Guarantee”}
Service Scope : SCOP	SCOP={“Number of Request: Max1000/Hour”}
Service Level Objective: SLO	SLO={“Service Commitment 1: 100% to 99.95%”,” Service Commitment 2: <99.95% to 99.9%”}
Action Guarantee: AG	AG={“Credit Request Time: In 30 Days”}
Penalties: PEN	PEN={“Service Credit1 :5%”, “Service Credit2 :10%”}
Optional Services : OS	OS={“Redeem Service Credit: To Cash ”}
Restrictions: RES	RES={This SLA does not apply to any: ”downtime, suspension, or termination of any services”}
Exclusions: EXC	EXC={“No Service Credit: If Contract Breached”}

Table 6-14: SLA from SP1 (File Storage Service Provider)

Basic Concept Element	**Basic Concept Element Value**
Agreement Name: AN	AN = {“Amazon-File Storage Server” SLA}
Agreement Template:ATEMP	ATEMP = {“No.AmazonFSS-6”}
Agreement Terms: *ATER*	*ATER* **={** “ Transaction Time : Means time to complete one transaction”, “ Latency : Round trip

	delay between request and response”**}**
Purpose : SLAPU	SLAPU = {”File Storage Server Contract”}
Service Provider: SP	SP = {“Amazon Web Services”}
Service Consumer: SC	SC = {“Mr. A.S”}
Third Parties: TP	TP={“Payment Processor: MasterCard”}
Signatory Party: SIP	SIP ={“Service Provider: Amazon Cloud Services”, “Service Consumer: Mr. A.S”}
Supporting Party=SUP	SUP ={“Third Party: Payment Processor”}
Agreement Initiator: AIN	AIN ={“Amazon Cloud Services”}
Agreement Responder: ARES	ARES={“Mr. A.S”}
Validity Period: VP	VP= (SD, ED)
Starting Date: SD	SD= (“10:00”, “30”,” March”, “2016”)
Ending Date: ED	ED= (“10:00”, “30”,” March”, “2017”)
Service: S	S={“File Storage Server”}
Inputs-to-Service: I	I={“ Maximum Storage Space: 10 TB”, “ Maximum Upload File Size : 10GB”, “ Files Archived : 90 Days”, “ External Hard Drive Backup : Required”, “File Sync : Required”}
Outputs-of-Service: O	O={“ Maximum Storage Space: 25 TB”, “ Maximum Upload File Size : 25GB”, “ Files Archived : 100 Days”, “ External Hard Drive Backup : Supported ”, “ File Sync: Supported ”, “Price: £700”}
Service Description Terms:SDT	SDT={“ Maximum Storage Space ”, “ Maximum Upload File Size ”, ” Files Archived ”, “ External Hard Drive Backup ”, “ File Sync”}
Service Properties:SPRO	SPRO={”File Server URL: ABC”, “Port:YYY”,” OS Support: Linux”}
SLA Parameter: SLAPAR	SLAPAR={”Transaction Time”, “Latency”}
Metric: MET	MET={“ Transaction Time : seconds”, “ Latency : Seconds”}

Measurement Directive:MD	MD={“Transaction Tim:1 to 15”, “:15 to 30”},
Function: FUN	FUN={“ Transaction Time -FUN”, ” Latency - FUN”}
Any Attribute:AA	AA={“Driver Support: ODBC”}
Obligations: OB	OB={“Service Commitments”, ”Service Credits”, ”Action Guarantee”}
Service Scope : SCOP	SCOP={“Number of Transactions: Max100/Hour”}
Service Level Objective: SLO	SLO={“Service Commitment 1: Equal to or greater than 99% but less than 99.9% ”,” Service Commitment 2: Less than 99% ”}
Action Guarantee: AG	AG={“Credit Request Time: In 45 Days”}
Penalties: PEN	PEN={“Service Credit1 :10%”, “Service Credit2 :25%”}
Optional Services : OS	OS={“Redeem Service Credit: To Cash Points ”}
Restrictions: RES	RES={This SLA does not apply to any: ”downtime, suspension, or termination of any services”}
Exclusions: EXC	EXC={“No Service Credit: If service contract cancelled early ”}

Step 4: Negotiation

In this use case example, the negotiation is not demonstrated, so this step is not followed.

Step 5: Execution

Based on the IT Manager’s requirements, the most suitable composite service plan based on two services i.e. composite plan 4 (SP_4:S_1 and SP_1:S_2) from Table 6-12, its execution can be initiated (SM:4) (Figure 5-3), relevant SLAs for the required services are Established (SLAM:3) (Figure 5-3) and SLAs being Deployed (SLAM:4) (Figure

5-3). The trigger for execution of the composite plan and establishment/deployment of SLAs is shown in Table 6-15.

Table 6-15: Execution of Filtered Composite Service Plan

Selected Composite Plan No.	Execute (Yes/No)	Establish and Deploy (SLAs)
4	Yes	Yes

Step 6: Termination

The execution of services used in a composite services plan will be due for termination on an explicitly defined date and time in the SLAs (SLAM:5) (Figure 5-3) and services will be stopped safely following Decommission of services (SM:4) (Figure 5-3). The trigger for Termination of services used in selected service Plan 4, is shown in Table 6-16.

Table 6-16: Termination of Filtered Composite Service Plan

Selected Composite Plan No.	Terminate (Yes/No)	Decommission
4	Yes	Yes

Step 7: QoS Monitoring

After the successful/unsuccessful completion of a service plan, IT Manager's experience of services used will be recorded in the form of QoS score into the QoS Database component (QoSD:1) (Figure 5-3) of SLAAgent. The QoS score will be calculated (QoSM:4) (Figure 5-3) with the help of FIS implemented in SLAAgent using monitoring tools (shown in Table 2-27). The Table 6-17 shows the sample QoS score generated for the services used by IT Manager based on three QoS Term metrics i.e. QoS Term-1, QoS Term-2 and QoS Term-3 and its QoS score is calculated using Fuzzy Inference System from mockup calculations given in Appendix-B, Table 10-3.

Table 6-17: QoS Score as feedback calculated by FIS in SLAAgent

Service Used	QoS Term-1 Range (0-10)	QoS Term-2 Range(0 to 10)	QoS Term-3 Range(0 to 10)	QoS Score
SP_4 (Web Hosting Service Provider)	6	7	9	7.07875
SP_1 (File Storage Service Provider)	7	6	7	6.141563

6.3 Results

The results of the use case example based on computational services scenario are given in Table 6-18, where the requirements of company's IT Manager are shown in one column of table, and the requirements fulfilled by proposed SLAAgent framework are shown in front of each requirement in the other column of table.

Table 6-18: Results from SLAAgent for IT Manager's Requirements

No.	IT Manager's Requirement	Requirement Fulfillment by SLAAgent
1	IT Manager required two Services (Web Hosting Server and File Storage Server)	IT Manager was given the option to select the name and number of services (shown in Table 6-3).
2	IT Manager wanted to provide the information for Web Hosting Server	IT Manager was given the option to provide the requirement information for Web Hosting Server (shown in Table 6-4).
3	IT Manager wanted to provide the information for File Storage Server	IT Manager was given the option to provide the requirement information for File Storage Server (shown in Table 6-5).
4	IT Manager wanted to give the requirement preferences for the services	IT Manager was given the option to provide the requirement preferences for the services (shown in Table 6-6).
5	IT Manager wanted to see all Web Hosting Services with QoS information	The result from SLAAgent for all available Web Hosting Servers providers with QoS information was given shown in Table 6-9.
6	IT Manager wanted to see all	The result from SLAAgent for all available File

	File Storage Services with QoS information	Storage Server providers with QoS information was given shown in Table 6-10.
7	IT Manager wanted to see all possible composite services plans with QoS information	The result from SLAAgent for all possible composite services plans were given in Table 6-11.
8	IT Manager wanted the filtered service plans according to his requirement preferences	The result from SLAAgent for filtered service plans according to IT Manager's preferences were given in Table 6-12.
9	IT Manager wanted to see the SLAs for the services involved into the filtered services plan.	For the most suitable composite service plan matching IT Manager's requirement preferences (i.e. composite plan 4) the Service Level Agreement of Web Hosting Service provider (SP_4) and File Storage Service provider (SP_1) is shown in Table 6-13 and Table 6-14 respectively.
10	IT Manager wanted to execute the filtered service plan according to his requirement preferences.	The SLAAgent provided the option to trigger the execution of the service plan (shown in Table 6-15).
11	IT Manager executed service plan required its completion stage.	The result from SLAAgent provided the trigger for the termination process (shown in Table 6-16).
12	IT Manager wanted the service provider QoS calculated with the help of Third Party Monitoring Tools for the services he used.	The result from SLAAgent, calculated the QoS of service providers with the help of Monitoring Tools (shown in Table 6-17) then stored in QoS Database.

6.4 Chapter Summary

A use case example on computational services was discussed in this chapter. The use case example was demonstrated using the proposed steps from SLAAgent framework in Chapter 5. The consumer requirements were fulfilled using SLAAgent framework, which covered the selection of services based on QoS score in composite services plan. The output of the composite services plans was filtered to match the consumer preferences, and then the triggers for execution and termination of the services were demonstrated. Finally, the usage experience of the services by consumer was recorded into QoS Database of the SLAAgent in the form of QoS score.

7. Evaluation

7.1 Introduction

The chapter presents an evaluation of the proposed SLAAgent framework. The assessment of the proposed framework involves assessment of the framework and its components, the results achieved from applying the proposed framework on use case example and appraisal of what degree did the solution (SLAAgent framework) worked. Finally, a comparative evaluation is given in this chapter which involves the comparison of the SLAAgent framework with similar approaches.

7.2 Assessment of SLAAgent and its Components

In this assessment, the SLAAgent framework, its individual components and the interaction among them is assessed.The assessment is discussed for the SLAAgent framework as general and its individual components separately including Controller(C), Requirement Processor(RP), Service Manager(SM), SLA Manager(SLAM), QoS Monitor(QoSM) and QoS Database (QoSD) which are shown in Figure 5-3.

The evaluation criteria for the SLAAgent framework and its components are:

1. For what problem the framework and its components are proposed for?
2. How the framework and its components are designed from the point of view of structure, management and monitoring?
3. How the framework and its components can be implemented?
4. How the framework and its components' solution works for the problem? At what degree the results are achieved applying framework and components to use case example?
5. What are the advantages and disadvantages of the framework and its components?

7.2.1 The SLAAgent Framework

1. The SLAAgent framework has been proposed to answer the Research Question: "How to structure and manage Service Level Agreements automatically and effectively for value added Quality of Service during Web service composition".

2. The SLAAgent framework has been designed to extend the Service Oriented Architecture (SOA), with an addition of SLAAgent among service provider, service consumer and UDDI.

3. The UDDI, service provider, service consumer and SLAAgent framework can be implemented in any Web service framework implementation which is based on SOA using Algorithms from (Section 0), because due to the use of SOA, the implementation can be done independent of any technology. More specifically, the implementation of the Fuzzy Inference method shown in Table 4-7 as mockup calculations using a spreadsheet program can be successfully re-used for the implementation purpose. The prototype implementation of SLAAgent framework has also been provided in Appendix-C. The UDDI can be implemented using UDDI4J [119] and jUDDI[3]. The SLAAgent can be implemented in the Apache Axis Web service Framework [2], while the service providers can implement their services accordingly to their business requirements by using any Web service framework implementation of their own choice. The SLAAgent framework should be implemented using Algorithms from (Section 0).

4. The SLAAgent framework designed for composition of services with QoS score using SLAs was simulated in the use case example given in Chapter 6, to fulfill the requirements of the service consumer. The results shown in Table 6-18 produced from SLAAgent for IT Manager's requirements fulfilled the consumer requirements and achievement of the required goals was demonstrated adequately.

5. The main advantage of the SLAAgent framework is considered as the centralized system, which helps to maintain the QoS of service providers on a single location point. It uses Fuzzy Inference method to deal with QoS calculation for different QoS term metrics. Once it is installed on a central location, it can be accessed easily from anywhere by service consumers and service providers. The expansion of the SLAAgent can be done easily due to its central location. The disadvantage of the SLAAgent framework can be for example if the central server is affected which is used for SLAAgent framework, then all the communication stops, until

the full system has recovered. The initial installation on a central point may require very powerful servers, so that they can manage the wide range of clients in the form of service consumers and service providers but they can be very expensive to buy.

7.2.2 Controller

1. The Controller (C) component of the SLAAgent framework is used for creating the communication between service consumers and service providers, also for communicating with UDDI. All the SLAAgent components communicate with each other via the Controller.

2. The Controller (C) is designed to remain active all the time for receiving, sending and scheduling the communication between components of SLAAgent and outside the SLAAgent with service providers, service consumers and UDDI.

3. The Controller (C) is the component of SLAAgent framework, and should be implemented as the sub component of the SLAAgent framework using any Web service framework implementation using Algorithms from (Section 5.5.4).

4. The Controller (C) component designed as a sub component of the SLAAgent framework facilitates to create the communication between service providers, service consumers, UDDI and also framework components. For the use case example taken from computational services scenario discussed in Chapter 6, the Controller receives requirements from consumer, then processes and passes the requirements among SLAAgent framework components, UDDI and service providers and hence helps to achieve the required goals easily.

5. The Controller (C) component is designed to work as a central component for the control flow of entire SLAAgent framework, and its interaction with service providers, service consumers and UDDI. Due to its central location, it reduces the complexity of communication between components, while due to workload of all the components of framework, the Controller can become slow due to dealing

multiple requests and responses related to different tasks. Hence the Controller needs to use synchronization and more computing power in order to control the massive traffic flow of communication between the components.

7.2.3 Requirement Processor

1. The Requirement Processor (RP) component of SLAAgent framework is used for processing the service consumer requirements. This component receives requests from service consumers for discovery of one or more services, QoS information for service providers and composite service plans according to particular preferences.

2. The Requirement Processor (RP) is designed to give the options to service consumers for selecting the type of services they require, getting the inputs from service consumers for required services along with preferences for the services. More specifically, the service consumers on front end (user interface) are connected with this Requirement Processor component, because this component is the entry point to SLAAgent for service consumers.

3. The Requirement Processor (RP) is the component of the SLAAgent framework, and should be implemented as the sub component of the SLAAgent framework using any Web service framework implementation using Algorithms from (Section 5.5.5).

4. The Requirement Processor (RP) designed as a sub component of SLAAgent framework facilitates to receive the requirements from service consumers, processing the requirements using different components of framework, receiving the outputs from components and finally returning the results to the service consumers. For the use case example taken from computation services scenario discussed in Chapter 6, the Requirement Processor receives the requirements from consumer, then processes the requirements, returns the output results to service consumer and helps to achieve the intended goals easily.

5. The Requirement Processor (RP) is designed to receive the requirements from service consumers outside the SLAAgent framework. All the other SLAAgent framework components are mostly used by this component. Receiving inputs requests from different types of service consumers using various user interfaces, the processing of their requirements by Requirement Processor can be a difficult task due to heterogeneous format of requirements. So, the consumer interfaces should be designed according to the requirement structure accepted by Requirement Processor.

7.2.4 Service Manager

1. The Service Manager (SM) component of SLAAgent framework is used for managing the services, composite service plans, filtering the composite service plans and executing them on request from service consumers.

2. The Service Manager (SM) is designed to deal with services and service providers, specifically creating the composite service plans in order to fulfil the service consumer requirements involving the request for multiple services. It also filters the created service plans according to specific requirement preferences from service consumers. The formation of composite service plans involves the cartesian product of services in order to provide the maximum possible service plans, and filters them according to service consumer preferences.

3. The Service Manager (SM) is component of SLAAgent framework and should be implemented as the sub component of the SLAAgent framework using any Web service framework implementation using Algorithms from (Section 5.5.6).

4. The Service Manager (SM) component as a sub component of the SLAAgent framework facilitates to create the composite service plans and to filter them according to service consumer preferences. For the use case example taken from computational services scenario discussed in Chapter 6, the Service Manager creates the composite service plans for consumer and then filters those service plans according to consumer preferences. The Service Manager then triggers the

execution of the service plan on the mutual agreement between consumer and service providers (for Web Hosting Server provider and File Storage Server provider) selected in the given example, and hence helps to achieve the consumer goals easily.

5. The Service Manager (SM) creates the composite service plans and filters them according to service consumer preferences. This component can provide a range of service plans, but due to large number of services discovered from UDDI, the formation of composite service plans can take more time and more computing resources, hence the optimized techniques should be used for creating the composite service plans in order to get the efficient and effective results using less computing resources in short time.

7.2.5 SLA Manager

1. The SLA Manager (SLAM) component of SLAAgent framework is used to structure and manage the SLAs. It facilitates to converts the conventional SLAs given from service providers into well structured and easily manageable SLAs for service consumers which also can help for better QoS monitoring purpose.

2. The SLA Manager (SLAM) is designed to structure and mange the SLAs using various basic concept elements from Chapter 3, and utilizes the refined SLA lifecycle stages shown in Table 2-20.

3. The SLA Manager (SLAM) is the component of SLAAgent framework and should be implemented as the sub component of the SLAAgent framework using any Web service framework implementation using Algorithms from (Section 5.5.7).

4. The SLA Manager (SLAM) designed as sub component of the SLAAgent framework facilitates to create new SLAs based on existing SLAs given from service providers. For the use case example taken from computational services scenario discussed in Chapter 6, the SLA Manager gets the SLAs from service

providers (Web Hosting Server providers and File Storage Server providers) and then helps to restructures the SLAs and presents it to the consumers in a managed form, and hence achieves the required goals easily.

5. The SLA Manager (SLAM) helps to structure and manage the SLAs. It creates new SLAs compatible for SLAAgent framework components by getting the SLAs from service providers. However, the SLAs provided from service providers can be given in any structure using any format according to their services, so it can become difficult to interpret and restructure the SLAs properly within the SLAAgent framework. If all the service providers produce the SLAs in uniquely identifiable format such as XML then, it can be easy to read and restructure by SLAAgent framework for efficient utilization and management, also the negotiation between service providers and service consumers can become easier with the help of SLAAgent framework.

7.2.6 QoS Monitor

1. The QoS Monitor (QoSM) component of SLAAgent framework is used for monitoring SLA violations, receiving feedback for service usage obtained with the help of monitoring tools e.g. from Table 2-27 and building the service provider reputation in the form of a QoS score for a service using Fuzzy Inference System based on particular QoS metrics.

2. The QoS Monitor (QoSM) is designed to monitor the SLAs, and calculate the QoS information by generating the service provider reputation as QoS score using Fuzzy Inference System defined in Chapter 4. The Fuzzy Inferencing Method is formalized to support the QoS terms metrics that can be defined within SLAs and also without the help of SLAs. The structured monitoring helps to manage the SLAs and related SLA lifecycle stages performed easier.

3. The QoS Monitor (QoSM) is component of SLAAgent framework, and should be implemented as the sub component of SLAAgent framework using any Web service framework implementation using Algorithms from (Section 5.5.8).

4. The QoS Monitor (QoSM) component designed as sub component of the SLAAgent framework facilitates to monitor the QoS and building the QoS information in the form of service provider reputation score for different services provided by them. For the use case example taken from computational services scenario discussed in Chapter 6, the QoS Monitor receives the feedback of services used with the help of monitoring tools and then builds the service provider reputation for service providers using Fuzzy Inference method based on different QoS Term metrics and helps achieve the required goal of QoS monitoring.

5. The QoS Monitor (QoSM) monitors the QoS and builds the service provider reputation as QoS score based QoS term metrics within and without SLA terms from service providers. The selection of different QoS term metrics for calculating the QoS for service providers proposed by this thesis (Table 2-43) are also detailed enough and classified for SLAs and non SLA terms. But the assessment and accuracy of information provided for the QoS term metrics involved from different sources such as monitoring tools' reports and non SLA metrics are difficult to get accurately, because of the dynamic and unpredictable nature of services.

7.2.7 QoS Database

1. The QoS Database component of SLAAgent framework is used for storing and retrieving the QoS information about the services and service providers in the form of service provider QoS score.

2. The QoS Database can be accessed by the SLAAgent framework only, it stores the QoS information in the form of Tables using Relational Database architecture, and the management of QoS Database should be made by corresponding Database Management System used by SLAAgent framework.

3. The QoS Database is the component of SLAAgent framework, and it should be implemented as the sub component of SLAAgent framework using any Web

service framework implementation with the help of any suitable Database Management System using Algorithms from (Section 5.5.9).

4. The QoS Database component designed as a sub component of the SLAAgent framework facilitates for storing and retrieving the QoS information required to the service consumers. For the use case example from Chapter 6, the QoS Database component stores and retrieves required QoS information needed to the consumer for composite services plans, and hence achieves the required goal easily.

5. The QoS Database works as a repository of QoS information used in SLAAgent framework created by any suitable Database Management System. The structure of data Tables normally used in QoS Database should be relational, but if the QoS information is needed for too many services and service providers, then more efficient Database Management System will be required for better results produced by SLAAgent framework.

7.3 Comparative Evaluation

In this section, a comparative evaluation is performed between the approach used by SLAAgent framework and other highly relevant approaches used for SLA elements, SLA lifecycle stages, QoS terms and QoS selection techniques.

In Table 7-1, the different approaches for SLA elements, SLA lifecycles and QoS terms are given short forms for easy naming conventions for comparison and analysis purpose in this section.

Table 7-1 : Short forms for Different Approaches

Approach Reference	Approach used For	Approach No. Assigned	Approach Short Form
[73]	SLA Elements/ SLA Lifecycle Stages	Approach-1	App-1
[18]	SLA Elements	Approach-2	App-2
[68]	SLA Elements	Approach-3	App-3

[128]	SLA Lifecycle Stages	Approach-4	App-4
[13]	SLA Lifecycle Stages	Approach-5	App-5
[123]	QoS Terms	Approach-6	App-6
[109]	QoS Terms	Approach-7	App-7
[57]	QoS Terms	Approach-8	App-8
[80]	QoS Terms	Approach-9	App-9
[20]	QoS Terms	Approach-10	App-10
[64]	QoS Terms	Approach-11	App-11

7.3.1 Comparative Evaluation of SLA Elements

The Table 7-2 shows the SLA Elements used by Approaches: App-1, App-2, App-3 and proposed SLAAgent framework separately in each column. The Table 7-3 shows the distribution of SLA elements divided into groups with percentage shown for their existence in single or in combined approaches along with total number of SLA elements in each. The Figure 7-1 shows the comparative analysis chart of SLA Elements used by other approaches and proposed SLAAgent framework. The blue bar shows total SLA Elements in each group used in single or combined approaches, the red bar shows the percentage of particular group of SLA elements out of 31 SLA Elements in single or in combined approaches, the green bar shows the group of SLA Elements that are common in single or in combined approaches.

Table 7-2 : Comparison of SLA Elements from Different Approaches

S.No	SLA Elements	App-1	App-2	App-3	SLAA gent
1	Agreement Context		X		
2	Agreement Terms		X		X
3	Agreement Initiator		X		X
4	Agreement Responder		X		X
5	Service Provider		X		X
6	Service Consumer		X		X
7	Agreement Template		X		X
8	Any Attribute		X		X
9	Service References		X		

10	Service Properties		X		X
11	Qualifying Condition		X		
12	Business Value List		X		
13	Service Object	X			
14	Signatory Party	X			X
15	Third Party/Supporting Party	X			X
16	SLA Parameter	X			X
17	Metric	X			X
18	Measurement Directive	X			X
19	Function	X			X
20	Obligations	X			X
21	Restrictions			X	X
22	Optional Services			X	X
23	Exclusions			X	X
24	Administration			X	
25	Service Definition/ Service Description Terms/Service Terms	X	X		X
26	Parties	X		X	X
27	Purpose/Agreement Name		X	X	X
28	Agreement Expiration Time/ Validity Period		X	X	X
29	Scope/Service Scope		X	X	X
30	Guarantee/Action Guarantee/ Guarantee Terms/Penalties	X	X	X	X
31	Service Level Objectives	X	X	X	X

Table 7-3 : Comparative Analysis of SLA Elements

Reference	Element Numbers	Total Elements	Percentage from 31 Elements	Elements Common In Approaches
App-2	(1-12)	12	38.7%	1
App-1	(13-20)	8	25.8%	1
App-3	(21-24)	4	12.9%	1
App-1, App-2	(25)	1	3.2%	2
App-1, App-3	(26)	1	3.2%	2
App-2, App-3	(27-29)	3	9.67%	2
App-1, App-2,	(30,31)	2	6.45%	3

App-3				
SLAAgent	(2-8,10,14-23, 25-31)	25	80.64 %	4

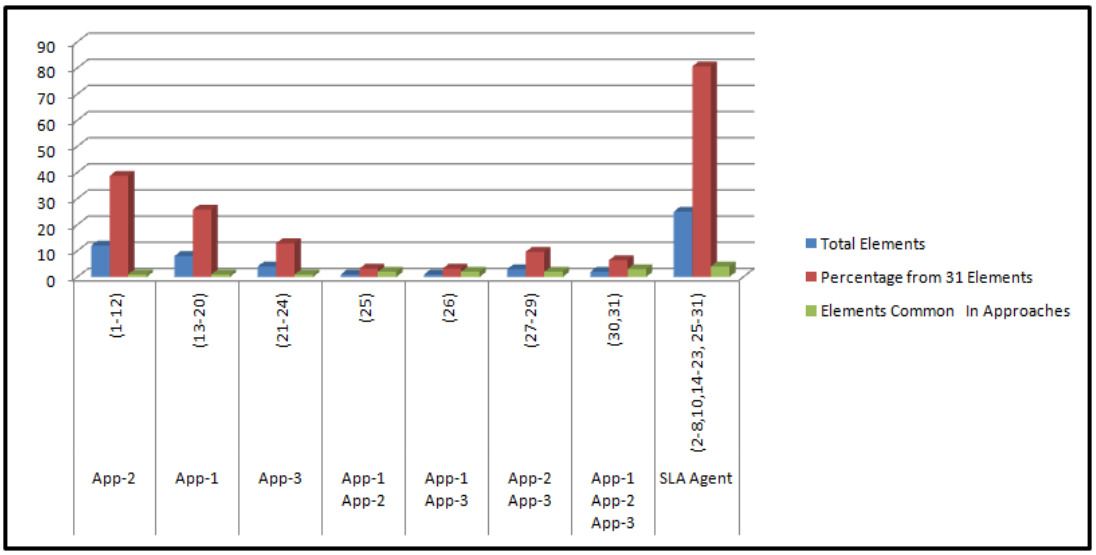

Figure 7-1 : Comparative Analysis Chart of SLA Elements

7.3.2 Comparative Evaluation of SLA Lifecycle Stages

The Table 7-4 shows the SLA lifecycle stages used by Approaches: App-1, App-4, App-5 and proposed SLAAgent framework separately in each column. The Table 7-5 shows the distribution of SLA lifecycle Stages divided into groups with percentage shown for their existence in single or in combined approaches along with total number of SLA lifecycle stages in each. The Figure 7-2 shows the comparative analysis chart of SLA lifecycle stages used by other approaches and proposed SLAAgent framework. The blue bar shows total SLA lifecycle stages in each group used in single or combined approaches, the red bar shows the percentage of particular group of SLA lifecycle Stages out of 12 SLA lifecycle Stages in single or in combined approaches, the green bar shows the group of SLA lifecycle Stages that are common in single or in combined approaches.

Table 7-4 : Comparison of SLA lifecycle Stages from Different Approaches

S.No	SLA lifecycle Stages	App-1	App-4	App-5	SLAAgent
1	Discover Service Provider		X		
2	SLA Deployment	X			X
3	Service Level Measurement	X			X
4	Service Development/ Preparation			X	X
5	Execution			X	X
6	Decommission			X	X
7	Define SLA / SLA Template Development		X	X	X
8	SLA Establishment/ Establish Agreement	X	X		X
9	Corrective Management Actions/ Enforce Penalties for SLA Violation	X	X		X
10	SLA Negotiation/Negotiation	X		X	X
11	Monitor SLA Violation/ Reporting/ Assessment	X	X	X	X
12	SLA Termination/ Terminate SLA/ Termination	X	X	X	X

Table 7-5 : Comparative Analysis of SLA Lifecycle Stages

Reference/ Short Form	Stage Numbers	Total Stages	Percentage from 12 Stages	Stages Common In Approaches
App-4	(1)	1	8.33%	1
App-1	(2-3)	2	16.66%	1
App-5	(4-6)	3	25%	1
App-4 App-5	(7)	1	8.33%	2
App-1 App-4	(8-9)	2	16.66%	2
App-1 App-5	(10)	1	8.33%	2
App-1 App-4 App-5	(11-12)	2	16.66%	3
SLAAgent	(2-12)	11	91.66%	4

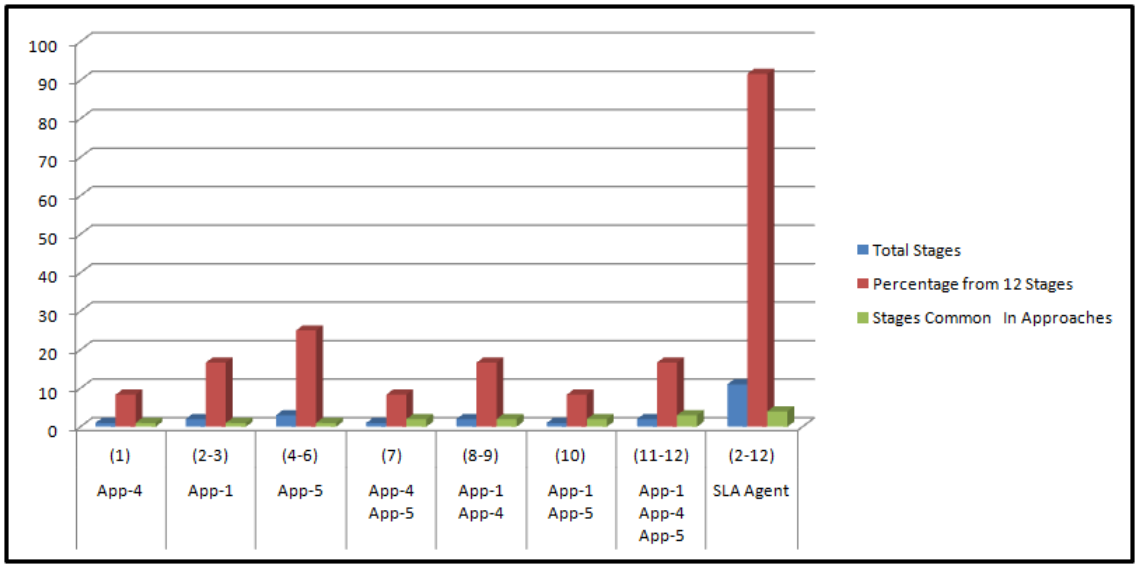

Figure 7-2 : Comparative Analysis Chart of SLA Lifecycle Stages

7.3.3 Comparative Evaluation of QoS Terms

The Table 7-6 shows the QoS terms used by Approaches: App-6, App-7, App-8, App-9, App-10, App-11 and proposed SLAAgent framework separately in each column. The Table 7-7 shows the distribution of QoS Attributes divided into groups with percentage shown for their existence in single or in combined approaches along with total number of QoS Attributes in each. The Figure 7-3 shows the comparative analysis chart of QoS terms used by other approaches and proposed SLAAgent framework. The blue bar shows total QoS terms in each group used in single or combined approaches, the red bar shows the percentage of particular group of QoS terms out of 35 QoS terms in single or in combined approaches, the green bar shows the group of QoS terms that are common in single or in combined approaches.

Table 7-6 : Comparison of QoS Terms from Different Approaches

No.	QoS Term	App-6	App-7	App-8	App-9	App-10	App-11	SLAAgent
1	Lifespan		X					X
2	Financial Status		X					X
3	Branches		X					X
4	Employees of Organization		X					X

5	No. Of Services		X					X
6	Brand Value		X					X
7	Success Rate		X					X
8	Advertising		X					X
9	Access Trust			X				X
10	Provision Trust			X				X
11	Certification Trust			X				X
12	Delegation Trust			X				X
13	Infrastructure Trust			X				X
14	Throughput				X			X
15	Response Time				X			X
16	Latency				X			X
17	Execution Time				X			X
18	Transaction Time				X			X
19	Authentication				X			X
20	Authorization				X			X
21	Accountability				X			X
22	Confidentiality				X			X
23	Traceability and Auditability				X			X
24	Non-Repudiation				X			X
25	Encryption				X			X
26	Availability					X	X	X
27	Accessibility					X	X	X
28	Accuracy					X	X	X
29	Reliability					X	X	X
30	Capacity					X	X	X
31	Scalability					X	X	X
32	Exception Handling (Stability)					X	X	X
33	Robustness (Flexibility)					X	X	X
34	Integrity(Data and Transaction)					X	X	X
35	Domain Attribute	X						X

Table 7-7 : Comparative Analysis of QoS Terms

Reference /Short Form	QoS Term Numbers	Total QoS Terms	Percentage from 35 QoS Terms	Terms Common In Approaches
App-7	(1-8)	8	22.85%	1
App-8	(9-13)	5	14.28%	1
App-9	(14-25)	12	34.28%	1
App-10 App-11	(26-34)	9	25.71%	2
App-6	(35)	1	2.85%	1
SLAAgent	(1-35)	35	100%	7

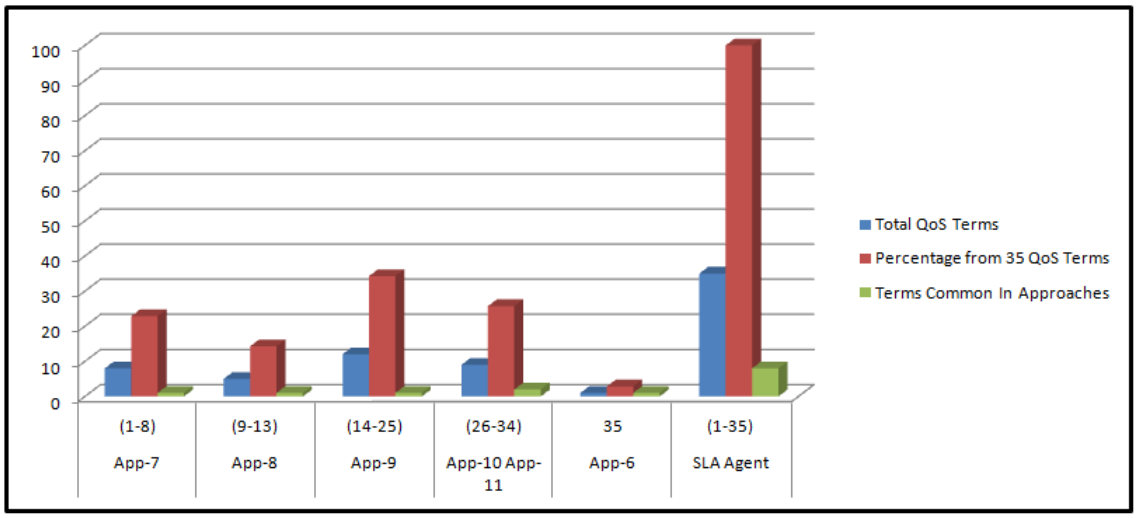

Figure 7-3 : Comparative Analysis Chart of QoS Terms

7.3.4 Comparative Evaluation of QoS Selection Techniques

This thesis has used Fuzzy Inference System as a Quality of Service measurement tool, which contributes to the research in an improved way of selecting services based on the Quality of Service with multiple QoS metric terms. An increased clarity of Service Level Agreements is also achieved for application of Fuzzy Logic to Quality of Service.

A discussion of different approaches using Fuzzy Inference System was presented in Chapter 2 (Section 2.5.7). The two most relevant approaches for comparison purpose

are Qu and Buyya [99] and Dastjerdi and Buyya [44]. These techniques have used Fuzzy Logic Inference System and are compared and contrasted with the FIS QoS selection technique used in the SLAAgent framework for Consumer requirements in Table 7-8, Service Management in Table 7-9, SLA Management in Table 7-10, QoS Monitoring in Table 7-11 and QoS Calculation Technique using FIS in Table 7-12.

It can be observed from Table 7-8 to Table 7-11 that the approaches used for comparison purpose with SLAAgent have more similarity of names and functionality, however the individual components proposed in the SLAAgent have greater detail and broader coverage of the aspects used for SLA management, monitoring, consumer requirements, service management and QoS calculation.

The comparision of the QoS selection technique used in the SLAAgent with other approaches given in Table 7-12 demonstrates the major difference of clarity of detailed SLAAgent structure with other FIS QoS selection techniques. The SLAAgent gives a more formalized representation of Fuzzy Inference Process, while the other approaches used in the comparison have skipped the details of the FIS process due to the third party FIS implementation tools used in their frameworks. The dynamic support for more than two inputs in SLAAgent using the derived formulas by this thesis shows an effective way of handling the problem. The decision making of FIS with the help of Fuzzy Associative Memory (FAM) based on experts and non-experts knowledge is a crucial achievement of the SLAAgent framework, while the similar support is not found in the other approaches during the comparison. Moreover the support for QoS terms within and without SLAs is also a vital achievement of the SLAAgent, which is more useful for utilizing the variety of QoS terms, while the other approaches have not concentrated to classify the QoS terms in the FIS process. The use of basic concept elements from Chapter 3, are also a great support for the overall SLAAgent framework functionality in order to increase the clarity in Service Level Agreements and with better implementation of Fuzzy Inference Process for QoS calculation.

Table 7-8: Compare and Contrast for Consumer Requirements

Qu and Buyya [99]	**Dastjerdi and Buyya** [44]	**SLAAgent**
Web Interface: this component • Allows users to enter the requirements • Describe the perception through this graphical interface	User Portal: this unit • Captures user's requirements	The framework component Requirement Processor (RP) : • Discovers service providers • service provider's QoS • Requests for composite services with QoS • Requests SLAs from service providers • Allows users to provide preferences over the services during selection for composite services plan

Table 7-9: Compare and Contrast for Service Management

Qu and Buyya [99]	**Dastjerdi and Buyya** [44]	**SLAAgent**
Discovery Service: this component : • Retrieves required services, with QoS information from Services repository	Composition Optimizer: • Builds possible compositions • Optimizes the compositions according to user preferences • Does decommissioning	The framework component Service Manager (SM) : • It composes the Services • Filters the composite service plan • Triggers for execution decommission of the services

Table 7-10: Compare and Contrast for SLA Management

Qu and Buyya [99]	**Dastjerdi and Buyya** [44]	**SLAAgent**
Trust Evaluation Service: this component : • Evaluates trust levels of functionality • Compares requirements with past trust benchmark results selects most suitable services	Discovery and Negotiation: • Tries to satisfy the QoS required to user by selecting the suitable service • Negotiates for sensible offers Failure Recovery: • Does failure recovery	The framework component SLA Manager (SLAM): • Defines SLA Templates • Supports SLA Negotiation • Establishes SLA • Deploys SLAs • Terminates SLAs • Enforces penalties for SLA Violation

Table 7-11: Compare and Contrast for QoS Monitoring

Qu and Buyya [99]	**Dastjerdi and Buyya** [44]	**SLAAgent**
Cloud benchmark service: • Continuously monitors performance of services using third party audit tools to ensure the integrity of provided services.	Monitoring and SLA Management: • Does health monitoring of deployed services	The framework component QoS Monitor (QoSM): • Keeps service level measurement • Monitors SLA violations • Receives feedback from users • Builds reputation in the form of QoS based on Fuzzy Inference System • Supports within SLA and without SLA QoS Metric Terms

Table 7-12: Compare and Contrast for QoS Calculation Technique Using FIS

Qu and Buyya [99]	**Dastjerdi and Buyya** [44]	**SLAAgent**
Formula for total number of rules not given (Rules dynamically created).	Formula for total number of rules not given.	Derives formula for total number of rules required (it depends on number of inputs and number of linguistic variables used in Fuzzy membership functions).
Formula for minimum and maximum number of rules being participated not given.	Formula for minimum and maximum number of rules being participated not given.	Derives formula for minimum and maximum number of rules participated for final output.
Formula for rules overlapping not given.	Formula for rules overlapping not given.	Derives formula for total number of rule overlapping.
Two inputs used, support for more number of inputs not shown.	Three inputs and one output used.	It is scalable, can support N number of inputs, where N is large (N>2).
FIS support for subgroups of inputs not given.	FIS support for subgroups of inputs not given.	FIS can be applied on N inputs or it can be applied on sub-groups made from N inputs.
Expert knowledge incorporated into the FAM not clearly	Partial Number of decisions given, the formulation of	Incorporates expert knowledge using Fuzzy Associative

mentioned.	Fuzzy Associative Memory based on expert knowledge not clearly mentioned.	Memory) for decision making.
formulation of Fuzzy Associative Memory based on calculation not clearly mentioned.	formulation of Fuzzy Associative Memory based on calculation not clearly mentioned.	Formulates decisions without expert knowledge for Fuzzy Associative Memory using custom calculation.
Simulation is shown on third party benchmark tool.	Implementation dependent on third party JFuzzyLogic.	Independent of any third party FIS Implementation.
QoS Based on SLA terms not mentioned explicitly.	Three QoS terms used, broader list of QoS terms not given.	Uses QoS metric terms within SLA and without SLA for QoS Calculation.
Formal Concept Elements not given for SLA, QoS.	Concept Elements for Provider, User, Request model given fully.	Uses Basic Concept Elements to increase the clarity of creating SLAs and QoS Term metrics.
Concept Elements for Fuzzy Inference Process not given.	Multi-objective algorithm implemented by third party jMetal framework.	Basic Concept Elements helps to formulate the Fuzzy Inference Process.

7.4 Chapter Summary

The chapter presented an assessment of the SLAAgent framework and its components based on the criteria setup. The chapter also discussed the comparision between SLAAgent and similar approaches for SLA elements, SLA lifecycle stages, QoS term metrics and QoS selection techniques by showing the comparison tables, comparative analysis tables and charts.

8. Conclusion

8.1 Introduction

In this chapter, the key findings/contributions of the thesis are summarized. The focus is put on the main research results that have been discussed, with the help of taking research directions from state of the art (Chapter 2). Starting from the survey of different research approaches relevant to this thesis, the focus was put on 3 highly relevant approaches for SLA elements (Section 2.3.1), 3 approaches for SLA lifecycles (Section 2.3.2), 6 approaches for QoS terms (Section 2.5.3) and various Multi-criteria Decision Making (MCDM) approaches for Quality of Service selection (Section 2.5.4) for further research. Moreover, the research question and its sub-problems introduced in Section 1.4 are revisited in this chapter and are critically analyzed to what extent those questions can be answered in this thesis. Finally, the thesis is concluded with an outlook on future research directions, which are derived from the aspects of the work that could not be sufficiently answered in the period of this thesis.

8.2 Review of Problem

Use of Service Level Agreements (SLAs) is crucial for business organizations to provide the value added goods or services to consumers to achieve the business goals successfully. SLAs also ensure the expected Quality of Service to consumers. This study investigates how efficient structural representation and management of SLAs can enhance the Quality of Service during composition of Web services.

The main research question in this thesis is: “How to structure and manage Service Level Agreements automatically and effectively for value added Quality of Service during Web service composition”.

The sub-problems related with the main research question addressed in this thesis include:

1. How to properly document the structure of SLAs
2. How to manage the SLAs
3. How to monitor the SLAs

4. How to calculate the Quality of Service
5. How to compose the services to fulfill consumer requirements based on QoS using SLAs.

Focusing on the main research question and its sub-problems, this thesis has introduced the SLAAgent framework for QoS calculation using Service Level Agreements in Web service composition.

8.3 Review of Solution

In this thesis, the elements of Service Level Agreements and metric terms related with Quality of Service calculation have been defined in detail with the help of basic concept elements from Chapter 3. Using basic concept elements, the Quality of Service is also calculated using Fuzzy Inference System that is defined in Chapter 4.

This thesis introduced an SLAAgent framework (QoS using SLAs) in Chapter 5. The framework is based on six components which includes Controller (C) (Section 5.3.1), Requirement Processor (RP) (Section 5.3.2), Services Manager (SM) (Section 5.3.3), SLA Manager (SLAM) (Section 5.3.4), QoS Monitor (QoSM) (Section 5.3.5) and QoS Database (QoSD) (Section 5.3.6). The high level structure of the SLAAgent framework is shown in Figure 5-2, and low level SLAAgent framework structure of its components is shown in Figure 5-3.

The Controller (C) is responsible for creating the communication between service providers, service consumers, Registry of Services (UDDI) and all SLAAgent framework components. The Requirement Processor (RP) receives the consumer requirements. The Services Manager (SM) deals with services and service providers for composing and preparing services for execution. The SLA Manager (SLAM) is responsible for SLA structures, SLAs management and SLAs negotiation between the service consumers and service providers. The QoS Monitor (QoSM) is responsible for maintaining, monitoring and calculating the QoS in the form of service provider QoS score on the basis of metric terms defined within SLAs and without SLAs. Fuzzy Information System process steps (Section 4.4) are used for calculating the QoS based

on different QoS terms. The QoS Database (QoSD) is responsible for storing and retrieving the individual QoS score of service providers.

Using SLAAgent, a use case example based on a scenario from computational services is worked out in Chapter 6. In the use case example the requirements of a consumer are processed by SLAAgent and then the corresponding results are discussed in Table 6-18.

8.3.1 Research Questions Revisited

Section 1.4 introduced the main research question and sub-problems related with main research question that directed to work on this thesis. In this section, main research question and its sub-problems are revisited. For the main research question and its sub-problems, summarized answers have been given with the help of proposed SLAAgent framework.

Research Question:

"*How to structure and manage Service Level Agreements automatically and effectively for value added Quality of Service during Web service composition****".***

SLAAgent framework is proposed for QoS calculations using SLAs in Web service composition which is discussed in Chapter 5, shown in Figure 5-2. The structure for SLAs, Quality of Service terms and management of SLAs is richly supported by basic concept definitions that are defined in Chapter 3. The SLAAgent framework has separate components for Requirements Processing (RP), Services Management (SM), SLA Management (SLAM), QoS Service Monitoring (QoSM) and QoS Database (QoSD) that are shown in Figure 5-3.

Sub-Problem 1:

How to properly document the structure of SLAs?

Using the detailed basic concept element definitions from Chapter 3 (Section 3.2) for SLA structures, the framework component SLA Manager (SLAM:1) (Figure 5-3) can document the SLA structures effectively and efficiently (Section 5.3.4.1).

Sub-Problem 2:

How to manage the SLAs?

The management of SLAs for its creations is carried out by (SLAM:1) (Figure 5-3), the negotiation of the existing SLAs is carried out by (SLAM:2) (Figure 5-3), the SLA establishment is carried out by (SLAM:3) (Figure 5-3), the SLA deployment is carried out by (SLAM:4) (Figure 5-3), the SLA termination is carried out by (SLAM:5) (Figure 5-3) and enforcement of penalties for SLA violations is carried out by (SLAM:6) (Figure 5-3), these sub-component tasks are defined in (Section 5.3.4).

Sub-Problem 3:

How to monitor the SLAs?

The monitoring of SLAs is carried out by (QoSM:2) (Figure 5-3) followed by Service Level Measurement (QoSM:1) (Figure 5-3), defined in Section 5.3.5 using monitoring tools from Table 2-27.

Sub-Problem 4:

How to calculate the Quality of Service?

The Quality of Service is calculated by sub-component (QoSM:4. Reputation Builder) (Figure 5-3) of the QoS Monitor component from SLAAgent (Section 5.3.5). Initially the feedback from service monitoring tools is received by (QoSM:3 Feedback Receiver) (Figure 5-3), then QoS is calculated using Fuzzy Inference System from (Section 4.4).

Sub-Problem 5:

How to compose the services to fulfill consumer requirements based on QoS using SLAs.

The output from the SLAAgent for the service consumer requirements results in many possible composite service plans (SM:2) (Figure 5-3). It is mandatory to filter the composite service plans in order to select the most appropriate service or set of services for a composite requirements, this is performed by Composition Filter (SM:3) (Figure

5-3), defined in (Section 5.3.3). Service composition is performed using (SM:2) (Figure 5-3) in terms of Quality of Service using SLAs for the consumer required services (Section 5.3.3.2).

In summary the sub- contributions as shown in Chapter 1 (Section 1.5) are:

1. Basic Concept Elements for SLA structures (Chapter 3, Section 3.2)
2. Framework Component for SLA Management (Chapter 5, Section 5.3.4)
3. Framework Component for QoS Monitoring using SLAs (Chapter 5, Section 5.3.5)
4. Quality of Service Calculations (Chapter 4, Section 4.4)
5. Framework Component for Managing Service Composition Using QoS Information from SLAs (Chapter 5, Section 5.3.3)
6. Advance in Fuzzy Inference Systems with more than two inputs (Chapter 4, Section 4.4)
7. Increased Clarity in Service Level Agreements (Chapter 2, Section 2.3.1)
8. Better Implementation of Fuzzy Inference System (Chapter 4, Section 4.6)
9. Improved way of selecting services based on QoS (Chapter 5, Section 5.3.2)
10. Application of Fuzzy Inference System to SLAs and QoS (Chapter 4, Section 4.5)
11. Usage of QoS Terms within and without SLA Parameters (Chapter 4. Section 4.6)

8.4 Future Work

The motivation of this research was to investigate the Role of Service Level Agreements in Web service Quality. The outcomes of the research have given a reasonable contribution to the state of art including Quality of Service calculation using Fuzzy Inference System.

For the future work, some ongoing issues that can usefully extend this work are given below:

1. For the discovery of service and service providers, the semantics web structures can be added to the proposed SLAAgent framework.

2. To explore the monitoring tools that do not have a strong SLAs and QoS relationship, and how they can benefit from the SLAAgent framework.

8.5 Chapter Summary

This chapter has summarized the key findings of the SLAAgent framework and has discussed how the research questions have been answered. This chapter reviewed the problems identified by the thesis and its solutions given by the thesis. Finally, the chapter ended with the future directions of the research.

References

[1] "Amazon CloudWatch", *URL: http://aws.amazon.com/cloudwatch/*, (Accessed: 02 November 2015).

[2] "Apache Axis, Web Services Project", *URL: http://ws.apache.org/axis*, (Accessed: 02 November 2015).

[3] "Apache JUDDI", *URL: http://juddi.apache.org*, (Accessed: 02 November 2015).

[4] "Azure Management Portal", *URL: http://azure.microsoft.com/en-us/documentation/articles/cloud-services-how-to-monitor/*, (Accessed: 02 November 2015).

[5] "Business Continuity and Business Compliance Terms", *URL: http://www.businessdictionary.com/definition/system-dependability.html#ixzz2XpuSs4U2*, (Accessed: 02 November 2015).

[6] "Fuzzy Inference Process", *URL: http://uk.mathworks.com/help/fuzzy/fuzzy-inference-process.html*, (Accessed: 02 November 2015).

[7] "Fuzzy Inference Systems", *URL: http://www.cs.princeton.edu/courses/archive/fall07/cos436/HIDDEN/Knapp/fuzzy004.htm*, (Accessed: 02 November 2015).

[8] "Google Cloud Monitoring", *URL: https://cloud.google.com/monitoring/*, (Accessed: 02 November 2015).

[9] "Grid Resource Allocation Agreement Protocol working group (GRAAP-WG)", *URL: http://forge.gridforum.org/sf/projects/graap-wg*, (Accessed: 02 November 2015).

[10] "HP Cloud Monitoring", *URL: http://www.hpcloud.com/products-services/monitoring*, (Accessed: 02 November 2015).

[11] "Open Grid Forum", *URL: http://www.ogf.org/*, (Accessed: 02 November 2015).

[12] "Organization for the Advancement of Structured Information Standards", *URL: https://www.oasis-open.org/*, (Accessed: 02 November 2015).

[13] "Telemanagement (TM) Forum. SLA Handbook Solution Suite V2.0 (2008)", *URL:*

http://www.tmforum.org/DocumentLibrary/SLAHandbookSolution/2035/Home.html, (Accessed: 02 November 2015).

[14] Adams, S., "ITIL V3 foundation handbook", *Stationery Office*, 2009.

[15] Aib, I., Agoulmine, N. and Pujolle, G., "The generalized service level agreement model and its application to the SLA driven management of wireless environments", *International Arab Conference on Information Technology*, 2004.

[16] Alabool, H. M. and Mahmood, A. K., "Trust-based service selection in public cloud computing using fuzzy modified VIKOR method", *Australian Journal of Basic and Applied Sciences*, vol. 7, no. 9, pp. 211-220, 2013.

[17] Andrews, T., Curbera, F., Dholakia, H., Goland, Y., Klein, J., Leymann, F. and Weerawarana, S., "Business process execution language for web services", 2003.

[18] Andrieux, A., Czajkowski, K., Dan, A., Keahey, K., Ludwig, H., Nakata, T. and Xu, M., "Web services agreement specification (WS-Agreement)", *Global Grid Forum*, vol. 2, 2004.

[19] Avila, S.D.G. and Djemame, K., "Fuzzy logic based qos optimization mechanism for service composition", *Service Oriented System Engineering (SOSE), IEEE 7th International Symposium*, pp. 182-191, 2013.

[20] Avizienis, A., Laprie, J. C. and Randell, B., "Fundamental concepts of dependability", *University of Newcastle upon Tyne, Computing Science, Technical Report N01145, LAASCNRS*, 2001.

[21] Baliyan, N. and Kumar, S., "Quality assessment of software as a service on cloud using fuzzy logic", *Cloud Computing in Emerging Markets (CCEM), IEEE International Conference*, pp. 1-6, 2013.

[22] Beek, M. T., Bucchiarone, A. and Gnesi, S., "A survey on service composition approaches: From industrial standards to formal methods. Technical Report 2006-TR-15.", *Technical Report 2006-TR-15. ISTI, Consiglio Nazionale delle Ricerche*, 2006.

[23] Blake, M. B., Cummings, D. J., Bansal, A. and Bansal, S. K., "Workflow composition of service level agreements for web services", *Decision Support Systems*, vol. 53, no. 1, pp. 234-244, 2012.

[24] Boley, H., "RuleML: The Rule Markup Initiative 2002", *URL: http://www.ruleml.org*, (Accessed: 02 June 2015).

[25] Booth, D., Haas, H., McCabe, F., Newcomer, E., Champion, M., Ferris, C. and Orchard, D., "Web Services Architecture, W3C Working Group Note 11 February 2004", *URL: http://www.w3.org/TR/ws-arch/*, (Accessed: 02 November 2015).

[26] Brans, J. P. and Vincke, P. , "A Preference Ranking Organisation Method: (The PROMETHEE Method for Multiple Criteria Decision-Making)", *Management science*, vol. 31, no. 6, pp. 647-656, 1985.

[27] Bratanis, K., Dranidis, D. and Simons, A. J., "Towards run-time monitoring of web services conformance to business-level agreements", *Testing-Practice and Research Techniques*, pp. 203-206, 2010.

[28] Bruijn, J.D., Bussler, C., Domingue, J., Fensel, D., Hepp, M., Kifer, M. and Stollberg, M., "Web Service Modeling Ontology (WSMO), W3C Member Submission 3 June 2005", *URL: http://www.w3.org/Submission/WSMO/*, (Accessed: 02 November 2015).

[29] Bruijn, J.D., Lausen, H., Polleres, A. and Fensel, D., "Web Service Modeling Language (WSML), W3C Member Submission 3 June 2005", *URL: http://www.w3.org/Submission/WSML/*, (Accessed: 02 November 2015).

[30] Büyüközkan, G. and Çifçi, G. , "A novel hybrid MCDM approach based on fuzzy DEMATEL, fuzzy ANP and fuzzy TOPSIS to evaluate green suppliers", *Expert Systems with Applications*, vol. 39, no. 3, pp. 3000-3011, 2012.

[31] Cannon, D., Cannon, D., Wheeldon, D., Lacy, S. and Hanna, A., "ITIL Service Strategy, TSO", 2011.

[32] Cardoso, J., "Semantic Web Services: Theory, Tools and Applications", *IGI Global*, 2007.

[33] Charnes, A., Cooper, W. W. and Ferguson, R. O., "Optimal estimation of executive compensation by linear programming", *Management science*, vol. 1, no. 2, 1955.

[34] Charnes, A., Cooper, W. W. and Rhodes, E. , "Measuring the efficiency of decision making units", *European journal of operational research*, vol. 2, no. 6,

pp. 429-444, 1978.

[35] Chen, W. C. and Johnson, A. L., "A unified model for detecting efficient and inefficient outliers in data envelopment analysis", *Computers & Operations Research*, vol. 37, no. 2, pp. 417-425, 2010.

[36] Chinnici, R., Moreau, J. J., Ryman, A. and Weerawarana, S., "Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language, W3C Recommendation 26 June 2007", *URL: http://www.w3.org/TR/wsdl20/*, (Accessed: 02 November 2015).

[37] Cho, J., Kwon, K. and Park, Y., "Q-rater: A collaborative reputation system based on source credibility theory", *Expert Systems with Applications*, vol. 36, no. 2, pp. 3751-3760, 2009.

[38] Cimpian, E. and Zaremba, M., "Web Service Execution Environment (WSMX), W3C Member Submission 3 June 2005", *URL: http://www.w3.org/Submission/WSMX/*, (Accessed: 02 November 2015).

[39] Clement, L., Hately, A., Riegen, C.V. and Rogers, T., "UDDI Version 3.0. 2 UDDI Spec Technical Committee Draft, Dated 2004-10-19, Organization for the Advancement of Structured Information Standards (OASIS)", *URL: http://www.uddi.org/pubs/uddi_v3.htm*, (Accessed: 02 November 2015).

[40] Comuzzi, M., Kotsokalis, C., Rathfelder, C., Theilmann, W., Winkler, U. and Zacco, G., "A framework for multi-level SLA management", *Service-Oriented Computing. ICSOC/ServiceWave 2009 Workshops, Springer Berlin Heidelberg*, pp. 187-196, 2010.

[41] Courbis, C. and Finkelstein, A., "Weaving aspects into web service orchestrations", *Proceedings of IEEE International Conference on Web Services, ICWS*, pp. 219-226, 2005.

[42] Cristóbal, J.R.S. , "Multi-criteria decision-making in the selection of a renewable energy project in spain: The Vikor method", *Renewable energy*, vol. 36, no. 2, pp. 498-502, 2011.

[43] Curbera, F., Duftler, M., Khalaf, R., Nagy, W., Mukhi, N. and Weerawarana, S. , "Unraveling the Web services web: an introduction to SOAP, WSDL, and UDDI", *Internet Computing, IEEE*, vol. 6, no. 2, pp. 86-93, 2002.

[44] Dastjerdi, A. V. and Buyya, R., "Compatibility-aware cloud service composition under fuzzy preferences of users", *Cloud Computing, IEEE Transactions*, vol. 2, no. 1, pp. 1-13, 2014.

[45] Dellarocas, C., "The digitization of word of mouth: Promise and challenges of online feedback mechanisms", *Management science*, vol. 49, no. 10, pp. 1407-1424, 2003.

[46] Deng, J.L., "Control problems of grey systems", *Systems and Control Letters*, vol. 1, no. 5, pp. 288-294, 1982.

[47] Feenstra, R. W., Janssen, M. and Wagenaar, R. W., "Evaluating web service composition methods: the need for including multi-actor elements", *The Electronic Journal of e-Government*, vol. 5, no. 2, pp. 153-164, 2007.

[48] Fensel, D. and Bussler, C., "The web service modeling framework WSMF", *Electronic Commerce Research and Applications*, vol. 1, no. 2, pp. 113-137, 2002.

[49] Frey, S., Luthje, C., Reich, C. and Clarke, N., "Cloud QoS Scaling by Fuzzy Logic", *Cloud Engineering (IC2E), IEEE International Conference*, pp. 343-348, 2014.

[50] Frolund, S. and Koisten, J., *Qml: A language for quality of service specification*.: Hewlett-Packard Laboratories, 1998.

[51] Frowe, I., "Professional trust", *British Journal of Educational Studies*, vol. 53, no. 1, pp. 34-53, 2005.

[52] Gabus, A. and Fontela, E., "Perceptions of the world problematique: Communication procedure, communicating with those bearing collective responsibility (DEMATEL Report No. 1)", *Battelle Geneva Research Centre, Geneva, Switzerland*, 1973.

[53] Gavade, R. K., "Multi-Criteria Decision Making: An overview of different selection problems and methods", *International Journal of Computer Science and Information Technologies*, vol. 5, no. 4, pp. 5643-5646, 2014.

[54] Geraci, A., Katki, F., McMonegal, L., Meyer, B., Lane, J., Wilson, P. and Springsteel, F., "IEEE standard computer dictionary: Compilation of IEEE standard computer glossaries", *IEEE Press*, 1991.

[55] Gil-Aluja, J. (Ed.)., "Handbook of management under uncertainty", *Springer Science & Business Media*, vol. 55, 2013.

[56] Governatori, G. and Milosevic, Z., "A formal analysis of a business contract language", *International Journal of Cooperative Information Systems*, vol. 15, no. 04, pp. 659-685, 2006.

[57] Grandison, T. and Sloman, M., "A survey of trust in internet applications", *Communications Surveys & Tutorials, IEEE*, vol. 3, no. 4, pp. 2-16, 2000.

[58] Gudgin, M., Hadley, M., Mendelsohn, N., Moreau, J. J. and Nielsen, H. F., "SOAP Version 1.2 Part 1: Messaging Framework (Second Edition), W3C Recommendation 27 April 2007", *URL: http://www.w3.org/TR/soap12-part1/*, (Accessed: 02 June 2015).

[59] Guidara, I., Chaari, T., Fakhfakh, K. and Jmaiel, M., "A comprehensive survey on intra and inter organizational agreements", *Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), IEEE 21st International Workshop*, pp. 411-416, 2012.

[60] Gutowska, A., Sloane, A. and Buckley, K. A., "On desideratum for B2C e-commerce reputation systems", *Journal of computer science and technology*, vol. 24, no. 5, pp. 820-832, 2009.

[61] Haq, I. U., Huqqani, A. A. and Schikuta, E., "Hierarchical aggregation of service level agreements", *Data & Knowledge Engineering*, vol. 70, no. 5, pp. 435-447, 2011.

[62] Harputlugil, T., Prins, M., Gültekin, A.T. and Topçu, Y.I., "Conceptual framework for potential implementations of multi criteria decision making (MCDM) methods for design quality assessment", *Management and Innovation for a Sustainable Built Environment (MISBE), CIB International Conference, Delft University of Technology Amsterdam*, 2011.

[63] Hasan, J. and Duran, M., "Expert Service-Oriented Architecture in C# 2005", *Dreamtech Press*, 2006.

[64] Hong-Linh, T., Samborski, R. and Fahringer, T., "Towards a framework for monitoring and analyzing QoS metrics of grid services", *e-Science and Grid Computing, e-Science'06, Second IEEE International Conference*, pp. 65-65,

2006.

[65] Hwang, C.L. and Yoon, K., "Multiple Attribute Decision Making Methods and Applications A State-of-the-Art Survey", *New York: Springer-Verlag*, vol. 186, 1981.

[66] Jadeja, Y., Modi, K. and Goswami, A., "Context Based Dynamic Web Services Composition Approaches: A Comparative Study", *International Journal of Information and Education Technology*, vol. 2, no. 2, pp. 164-166, 2012.

[67] Jaiganesh, M. and Kumar, A.V.A, "B3: fuzzy-based data center load optimization in cloud computing", *Mathematical Problems in Engineering*, 2013.

[68] Jin, L. J., Machiraju, V. and Sahai, A., "Analysis on service level agreement of web services", *Technical Report HPL-2002-180, HP Laboratories, Palo Alto, CA.*, 2002.

[69] Jøsang, A. and Golbeck, J., "Challenges for robust trust and reputation systems", *Proceedings of the 5th International Workshop on Security and Trust Management (SMT 2009), Saint Malo, France*, 2009.

[70] Jøsang, A., Ismail, R. and Boyd, C., "A survey of trust and reputation systems for online service provision", *Decision support systems*, vol. 43, no. 2, pp. 618-644, 2007.

[71] Juric, M. B., "A Hands-on Introduction to BPEL", *Oracle (white paper)*, 2006.

[72] Kavantzas, N., Burdett, D., Ritzinger, G., Fletcher, T., Lafon, Y. and Barreto, C., "Web Services Choreography Description Language Version 1.0, W3C Candidate Recommendation 9 November 2005", *URL: http://www.w3.org/TR/ws-cdl-10/*, (Accessed: 02 November 2015).

[73] Keller, A. and Ludwig, H., "The WSLA framework: Specifying and monitoring service level agreements for web services", *Journal of Network and Systems Management*, vol. 11, no. 1, pp. 57-81, 2003.

[74] Knublauch, H., Fergerson, R. W., Noy, N. F. and Musen, M. A., "The Protégé OWL plugin: An open development environment for semantic web applications", *The Semantic Web–ISWC 2004*, *Springer Berlin Heidelberg*, pp. 229-243, 2004.

[75] Kosko, B., "Global stability of generalized additive fuzzy systems", *Systems,*

Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions, vol. 28, no. 3, pp. 441-452, 1998.

[76] Kotsokalis, C. and Winkler, U., "Translation of service level agreements: a generic problem definition", *Service-Oriented Computing. ICSOC/ServiceWave 2009 Workshops*, pp. 248-257, 2010.

[77] Kozlenkov, A. and Schroeder, M., "PROVA: Rule-based Java-scripting for a bioinformatics semantic web", *Data Integration in the Life Sciences*, pp. 17-30, 2004.

[78] Kurbel, K. E., "The making of information systems: software engineering and management in a globalized world", *Springer Science and Business Media*, 2008.

[79] Lamanna, D. D., Skene, J. and Emmerich, W., "SLAng: a language for defining service level agreements", *proceedings of 9th IEEE International Workshop on Future Trends of Distributed Computing Systems (FTDCS 2003), San Juan, Puerto Rico, 28-30, IEEE Computer Society*, pp. 100-106, 2003.

[80] Lee, K., Jeon, J., Lee, W., Jeong, S. H. and Park, S. W., "QoS for Web Services: Requirements and Possible Approaches, W3C Working Group Note 25 November 2003", *URL : http://www.w3c.or.kr/kr-office/TR/2003/ws-qos*, (Accessed: 02 June 2015).

[81] Liu, L., "Systematic measurement of centralized online reputation systems 2011", *Doctoral dissertation, Durham University, URL: http://etheses.dur.ac.uk/881/*, (Accessed: 02 November 2015).

[82] Ludwig, A. and Franczyk, B., "Managing dynamics of composite Service Level Agreements with COSMA", *Fuzzy Systems and Knowledge Discovery, FSKD'08, Fifth International Conference*, pp. 584-589, 2008.

[83] Mamdani, E. H. and Assilian, S., "An experiment in linguistic synthesis with a fuzzy logic controller", *International journal of man-machine studies*, vol. 7, no. 1, pp. 1-13, 1975.

[84] Manola, F., Miller, E. and McBride, B., "RDF Primer, W3C Recommendation 10 February 2004", *URL: http://www.w3.org/TR/rdf-primer/*, (Accessed: 02 November 2015).

[85] Marquis, H., "A Study Guide to Service Catalogue from the Principles of Itil V3",

APMG International Service Catalogue, 2010.

[86] Martin, D., Burstein, M., Hobbs, J., Lassila, O., McDermott, D., McIlraith, S. and Sycara, K., "OWL-S: Semantic Markup for Web Services, W3C Member Submission 22 November 2004", *http://www.w3.org/Submission/OWL-S/*, (Accessed: 02 November 2015).

[87] Mell, P. and Grance, T., "The NIST definition of cloud computing, recommendations of the national institute of standards and technolog", *National Institute of Standards and Technology*, pp. 800-145, 2011.

[88] Mohammadshahi, Y., "A state-of-art survey on TQM applications using MCDM techniques", *Decision Science Letters*, vol. 2, no. 3, pp. 125-134, 2013.

[89] Molina-Jimenez, C., Shrivastava, S., Solaiman, E. and Warne, J., "Run-time monitoring and enforcement of electronic contracts", *Electronic Commerce Research and Applications*, vol. 3, no. 2, pp. 108-125, 2004.

[90] Newsome, J., Shi, E., Song, D. and Perrig, A., "The sybil attack in sensor networks: analysis & defenses", *Proceedings of the 3rd international symposium on Information processing in sensor networks*, pp. 259-268, 2004.

[91] Novák, V., Perfilieva, I. and Mockor, J., "Mathematical principles of fuzzy logic", *Springer Science and Business Media*, 2012.

[92] Opricovic, S. and Tzeng, G. H., "Compromise solution by MCDM methods: A comparative analysis of VIKOR and TOPSIS", *European Journal of Operational Research*, vol. 156, no. 2, pp. 445-455, 2004.

[93] Padgett, J., Djemame, K. and Dew, P., "Grid service level agreements combining resource reservation and predictive run-time adaptation", *Proc. of the UK e-Science All Hands Meeting, Nottingham, UK*, 2005.

[94] Padgett, J., Djemame, K. and Dew, P., "Grid-based SLA management", *Advances in Grid Computing-EGC, Springer Berlin Heidelberg*, pp. 1076-1085, 2005.

[95] Paschke, A., "RBSLA A declarative Rule-based Service Level Agreement Language based on RuleML", *Computational Intelligence for Modelling, Control and Automation, 2005 and International Conference on Intelligent Agents, Web Technologies and Internet Commerce*, vol. 2, pp. 308-314, 2005.

[96] Paschke, A., Bichler, M. and Dietrich, J. , "Contractlog: An approach to rule based monitoring and execution of service level agreements", *Rules and rule markup languages for the semantic web, Springer Berlin Heidelberg*, pp. 209-217, 2005.

[97] Pati, R. K., Vrat, P., and Kumar, P. , "A goal programming model for paper recycling system", *Omega*, vol. 36, no. 3, pp. 405-417, 2008.

[98] Prasath, V., Buvanesvari, R., Anitha, V. and Keerthana, M., "Improving web service selection using fuzzy quality of protection", *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, vol. 3, no. 9, pp. 116-120, 2014.

[99] Qu, C. and Buyya, R., "A cloud trust evaluation system using hierarchical fuzzy inference system for service selection", *Advanced Information Networking and Applications (AINA), IEEE 28th International Conference*, pp. 850-857, 2014.

[100] Ran, S., "A model for web services discovery with QoS", *ACM Sigecom exchanges*, vol. 4, no. 1, pp. 1-10, 2003.

[101] Rehman, Z.U., Hussain, O. K. and Hussain, F. K., "IAAS Cloud Selection using MCDM Methods", *IEEE 10th International Conference on e-Business Engineering, Los Alamitos, CA, USA*, pp. 246-251, 2012.

[102] Resnick, P., Kuwabara, K., Zeckhauser, R. and Friedman, E., "Reputation systems", *Communications of the ACM*, vol. 43, no. 12, pp. 45-48, 2000.

[103] Roy, B. and Vanderpooten, D., "The European school of MCDA: Emergence, basic features and current works", *Journal of Multi-Criteria Decision Analysis*, vol. 5, no. 1, pp. 22-38, 1996.

[104] Saaty, T. L., "The analytic network process", *Iranian Journal of Operations Research*, vol. 1, no. 1, pp. 1-27, 2008.

[105] Saaty, T. L., "The analytic network process: decision making with dependence and feedback", *The organization and prioritization of complexity*, 1996.

[106] Saaty, T. L., "What is the analytic hierarchy process?", *Springer Berlin Heidelberg*, pp. 109-121, 1988.

[107] Sahai, A., Durante, A. and Machiraju, V., "Towards automated SLA management

for web services", *Hewlett-Packard Research Report HPL-2001-310 (R. 1)*, 2002.

[108] Samant, R., Deshpande, S. and Jadhao, A., "Survey on Multi Criteria Decision Making Methods", *International Journal of Innovative Research in Science, Engineering and Technology*, vol. 4, no. 8, pp. 7175-7178, 2015.

[109] Sha, M. M. and Vivekanandan, K., "Selection of Web Services Based on Provider's Reputation", *International Journal of Innovative Technology and Exploring Engineering (IJITEE) ISSN: 2278-3075*, vol. 2, no. 3, 2013.

[110] Shaikh, A., Memon, M., Memon, N. and Misbahuddin, M., "The role of service oriented architecture in telemedicine healthcare system", *International Conference on Complex, Intelligent and Software Intensive Systems, CISIS'09*, pp. 208-214, 2009.

[111] Srinivasan, L. and Treadwell, J., "An overview of service-oriented architecture, web services and grid computing", *HP Software Global Business Unit*, vol. 2, 2005.

[112] Srivastava, B. and Koehler, J., "Web service composition-current solutions and open problems", *ICAPS 2003 workshop on Planning for Web Services*, vol. 35, pp. 28-35, 2003.

[113] Sugeno, M., "Industrial applications of fuzzy control", *Elsevier Science Inc.*, 1985.

[114] Sumra, R. and Arulazi, D., "Quality of service for web services-demystification, limitations, and best practices March 4, 2003", *URL: http://www.developer.com/services/article.php/2027911*, (Accessed: 02 November 2015).

[115] Tahriri, F. and Taha, Z., "Critical success factors for technology selection specifically ROBOTS", *Journal of Business Management and Economics*, vol. 2, no. 1, pp. 89-97, 2011.

[116] Tebbani, B. and Aib, I., "GXLA a language for the specification of service level agreements", *Autonomic Networking, Springer Berlin Heidelberg*, pp. 201-214, 2006.

[117] Theilmann, W., Happe, J., Kotsokalis, C., Edmonds, A., Kearney, K. and Lambea, J., "A reference architecture for multi-level sla management", *Journal of*

Internet Engineering, vol. 4, no. 1, pp. 289-298, 2010.

[118] Theilmann, W., Winkler, U., Happe, J. and Abril, I. M.D, "Managing on-demand business applications with hierarchical service level agreements", *Future Internet-FIS*, pp. 97-106, 2010.

[119] Tidwell, D., "UDDI4J: Matchmaking for Web services Interacting with a UDDI server", *IBM developerWorks*, 2001.

[120] Tosic, V., Pagurek, B. and Patel, K., "WSOL- A language for the formal specification of classes of service for web services", *International Conference on Web Services (ICWS)*, pp. 375-381, 2003.

[121] Tsukamoto, Y., "An approach to fuzzy reasoning method", *Advances in fuzzy set theory and applications*, vol. 137, p. 149, 1979.

[122] Velasquez, M. and Hester, P. T., "An analysis of multi-criteria decision making methods", *International Journal of Operations Research*, vol. 10, no. 2, pp. 56-66, 2013.

[123] Wang, Y. and Vassileva, J., "Toward trust and reputation based web service selection: A survey", *International Transactions on Systems Science and Applications*, vol. 3, no. 2, pp. 118-132, 2007.

[124] Wei, G. W., "Gray relational analysis method for intuitionistic fuzzy multiple attribute decision making", *Expert systems with Applications*, vol. 38, no. 9, pp. 11671-11677, 2011.

[125] Whaiduzzaman, M., Gani, A., Anuar, N. B., Shiraz, M., Haque, M. N., and Haque, I. T., "Cloud service selection using multicriteria decision analysis", *The Scientific World Journal*, 2014.

[126] Whaiduzzaman, M., Haque, M. N., Rejaul Karim Chowdhury, M., and Gani, A. , "A study on strategic provisioning of cloud computing services", *The Scientific World Journal*, 2014.

[127] Winkler, M., Springer, T. and Schill, A., "Automating composite SLA management tasks by exploiting service dependency information", *Web Services (ECOWS), IEEE 8th European Conference*, pp. 59-66, 2010.

[128] Wustenhoff, E., "Service level agreement in the data center", *Sun Microsystems*

Professional Services, Sun BluePrints, April 2002, URL: http://www.walker-institute.ac.uk/~swsellis/tech/solaris/performance/doc/blueprints/0402/sla.pdf, (Accessed: 02 November 2015).

[129] Yang, Y. P. O., Shieh, H. M., Leu, J. D., and Tzeng, G. H., "A novel hybrid MCDM model combined with DEMATEL and ANP with applications", *International Journal of Operations Research*, vol. 5, no. 3, pp. 160-168, 2008.

[130] Zadeh, L. A., "Fuzzy sets", *Information and control*, vol. 8, no. 3, pp. 338-353, 1965.

[131] Zarandi, M. H. F., Mohammadhasan, N. and Bastani, S. , "A fuzzy rule-based expert system for evaluating intellectual capital", *Advances in Fuzzy Systems*, vol. 7, 2012.

9. Appendix-A

9.1 Lists of Cloud Computing Service Providers

The lists of services from most popular Cloud computing service providers are given in Table 9-1, Table 9-2, Table 9-3and Table 9-4.

Table 9-1: List of Cloud Computing Services from Amazon

Name of Service	Type of Service	IaaS/PaaS /SaaS	SLA Source
Amazon EC2	Compute	IaaS	http://aws.amazon.com/ec2/sla/
Auto Scaling	Compute	IaaS	
Elastic Load Balancing	Compute	IaaS	
Amazon WorkSpaces		IaaS	
Amazon S3 (Simple Storage Service)	Storage	IaaS	http://aws.amazon.com/s3/sla/
Amazon Glacier	Storage	IaaS	
AWS Storage Gateway	Storage	IaaS	
Amazon EBS (Elastic Block Store)	Storage	IaaS	
Amazon DynamoDB	Database	PaaS	
Amazon RDS (Relational Database Service)	Database	PaaS	http://aws.amazon.com/rds/sla/
Amazon Redshift	Database	PaaS	
Amazon ElastiCache	Database	PaaS	
Amazon VPC (Virtual Private Cloud)	Networking	IaaS	
Amazon Route 53	Networking	IaaS	http://aws.amazon.com/route53 /sla/
Amazon CloudFront	Networking	IaaS	
AWS Direct Connect	Networking	IaaS	
Amazon EMR (Elastic MapReduce)	Analytics	SaaS	
Amazon Kinesis	Analytics	SaaS	
Amazon Redshift	Analytics	SaaS	
AWS Data Pipeline	Analytics	SaaS	
Amazon AppStream	Application Services	SaaS	
Amazon CloudSearch	Application Services	SaaS	
Amazon SWF (Simple Workflow Service)	Application Services	SaaS	

Amazon SES (Simple Email Service)	Application Services	SaaS	
Amazon SNS (Simple Notification Service)	Application Services	SaaS	
Amazon SQS (Simple Queue Service)	Application Services	SaaS	
Amazon Elastic Transcoder	Application Services	SaaS	
AWS Identity and Access Management (IAM)	Deployment & Management	SaaS	
Amazon CloudWatch	Deployment & Management	SaaS	
AWS Elastic Beanstalk	Deployment & Management	SaaS	
AWS CloudFormation	Deployment & Management	SaaS	
AWS Data Pipeline	Deployment & Management	SaaS	
AWS OpsWorks	Deployment & Management	SaaS	
AWS CloudHSM	Deployment & Management	SaaS	
AWS CloudTrail	Deployment & Management	SaaS	

Table 9-2: List of Cloud Computing Services from Google

Name of Service	Type of Service	IaaS/PaaS /SaaS	SLA Source
Compute Engine	Compute	IaaS	https://developers.google.com/compute/sla
App Engine	Compute	IaaS	https://developers.google.com/appengine/sla
Cloud Storage	Storage	IaaS	https://developers.google.com/storage/sla
Cloud SQL	Storage/Database	PaaS	https://developers.google.com/cloud-sql/sla
Cloud Datastore	Storage/Database	PaaS	https://developers.google.com/datastore/sla
BigQuery	Big Data	PaaS	https://developers.google.com/bigquery/sla
Prediction API	Services	SaaS	https://developers.google.com/storage/sla
Translate API	Services	SaaS	
Cloud DNS	Services	SaaS	
Cloud Endpoints	Services	SaaS	

Table 9-3: List of Cloud Computing Service from Windows Azure

Name of Service	Type of Service	IaaS/PaaS /SaaS	SLA Source
Virtual Machines	Compute	IaaS	http://go.microsoft.com/fwlink/?linkid=296425&clcid=0x409
Web Sites	Compute	IaaS	http://go.microsoft.com/fwlink/?linkid=301329&clcid=0x409
Mobile Services	Compute	IaaS	http://go.microsoft.com/fwlink/?linkid=301328&clcid=0x409
Cloud Services	Compute	PaaS	http://go.microsoft.com/fwlink/?linkid=296425&clcid=0x409
Storage	Data Services	IaaS	http://go.microsoft.com/fwlink/p/?linkid=159705&clcid=0x409
SQL Database	Data Services	IaaS	http://go.microsoft.com/fwlink/p/?linkid=159706&clcid=0x409
HDInsight	Data Services	IaaS	http://go.microsoft.com/fwlink/?linkid=324496&clcid=0x409
Cache	Data Services	IaaS	http://go.microsoft.com/fwlink/p/?linkid=159707&clcid=0x409
Backup	Data Services	IaaS	http://go.microsoft.com/fwlink/p/?linkid=285729&clcid=0x409

Recovery Manager/ Recovery Services/ Hyper-V Recovery Manager	Data Services	IaaS	http://go.microsoft.com/fwlink/?linkid=386508&clcid=0x409
Media Services	App Services	SaaS	http://go.microsoft.com/fwlink/?linkid=268074&clcid=0x409
Service Bus	App Services	SaaS	http://go.microsoft.com/fwlink/?linkid=316511&clcid=0x409
Notification Hubs	App Services	SaaS	
Scheduler	App Services	SaaS	http://go.microsoft.com/fwlink/?linkid=391423&clcid=0x409
Automation	App Services	SaaS	
BizTalk Services	App Services	SaaS	http://go.microsoft.com/fwlink/?linkid=285728&clcid=0x409
Visual Studio Online	App Services	SaaS	http://go.microsoft.com/fwlink/?linkid=391424&clcid=0x409
Active Directory	App Services	SaaS	?
Multi-Factor Authentication	App Services	SaaS	http://go.microsoft.com/fwlink/?linkid=313266&clcid=0x409
API Management	App Services	SaaS	
Azure RemoteApp	App Services	SaaS	
Express Network/ Express Route	Network	SaaS	http://go.microsoft.com/fwlink/?linkid=391425&clcid=0x409
Virtual Network	Network	IaaS	http://go.microsoft.com/fwlink/?linkid=296425&clcid=0x409
Traffic Manager	Network	IaaS	http://go.microsoft.com/fwlink/?linkid=328187&clcid=0x409
CDN	Network	IaaS	http://go.microsoft.com/fwlink/p/?linkid=195943&clcid=0x409

Table 9-4: List of Cloud Computing Services from HP

Name of Service	**Type of Service**	**IaaS/PaaS /SaaS**	**SLA Source**
HP Cloud Compute	Compute	IaaS	http://www.hpcloud.com/sla/compute
HP Cloud Block Storage	Compute	IaaS	http://www.hpcloud.com/sla/block-storage
HP Cloud Object Storage	Storage	IaaS	http://www.hpcloud.com/sla/object-storage
HP Cloud Object Storage Request	Storage	PaaS	
HP Cloud CDN Bandwidth	Network	IaaS	http://www.hpcloud.com/sla/cdn
HP Cloud Non-CDN Bandwidth	Network	IaaS	
HP Cloud DNS	Network	IaaS	http://www.hpcloud.com/sla/dns
HP Cloud Relational Database	Database	PaaS	

9.2 SLAs from Cloud Computing Service Providers

The selected SLAs from HP and Amazon Cloud computing service providers are given in Table 9-5 and Table 9-6 respectively.

Table 9-5: Service Level Agreement from HP Cloud Compute

Service Commitment:

HP commits that HP Cloud Compute will be available 99.95% or more of the time in a given calendar month. If we (HP) fail to meet this commitment, just let us know and we will apply a service credit to your account. The service credit applied will be calculated by multiplying a) your total charges for HP Cloud Compute in a given Region during the month we failed to meet the commitment by b) the percentage credit you qualify for in the table below:

Monthly Availability % (per Region)	Credit to Bill for HP Cloud Compute for a Given Region (Not Total Bill)
100% to 99.95%	N/A
<99.95% to 99.9%	5%
<99.9% to 99.5%	10%
<99.5% to 99%	20%
<99.0%	30%

Definitions:

HP Cloud Compute refers to HP's compute service, and does not refer to peripheral or separate services, including but not limited to: the HP Cloud management console, HP Cloud language bindings, HP Cloud command line tools, HP Cloud CDN, HP Cloud Block Storage, or HP Cloud Object Storage. An "instance" means a customer's virtual machine created within HP Cloud Compute. A "Region" represents a geographic area that is no more than 100 miles in diameter and consists of multiple physically separate Availability Zones. An "Availability Zone" is a deployment of HP Cloud Compute which consists of a separate API endpoint in which customers can choose to create instances. "Monthly Availability %" is calculated per Region on a monthly basis, as 100% minus: i) Total instance-downtime-minutes, divided by ii) Total instance-minutes "Total instance-minutes" is defined as the

aggregate amount of time all instances are running for a customer during a given month in a given Region. "Total instance-downtime-minutes" is calculated as the sum of each instance's downtime minutes, during the course of a month. For each instance, "downtime minutes" are accrued starting at the beginning of the first 6 minute interval during which the instance was inaccessible and the user was unable to launch a replacement instance in the same Region, and continue until the ability to launch a replacement instance is restored, including the time that would be required for a replacement instance to become accessible.

- "Inaccessible" means that the operating system in the replacement instance could not respond to API or network requests, despite proper security group configuration, for 6 minutes or more. "Accessible" means that the operating system in the replacement instance could respond to network requests.
- "Unable to launch a replacement instance in the same Region" means that a request was sent to each HP Cloud Compute API endpoint for that Region but no replacement instance actually started and became accessible.

Any Region in which a customer has no HP Cloud Compute activity, defined as having 0 "total instance-minutes" on their bill in a given month, will be deemed to have had 100% availability for that customer for the given month. To be eligible for a service credit a customer must be running or trying to run instances in more than one Availability Zone within a Region during the period of time when the customer's instances were inaccessible.

Exclusions:

You are not entitled to a service credit if you are in breach of your Customer Agreement with HP, including your payment obligations. The inability to launch new instances due to exceeding your account quotas or improperly formed API requests are not covered by this SLA. To receive a service credit, you must file for a credit within 30 days following the end of the month in which availability was not met by contacting HP via the Contact Us link on the HP Cloud website with a description of the downtime, how you were affected, and for how long. HP reserves the right to withhold credit if it cannot verify the downtime or you cannot show that you were adversely affected in any way as a result of the downtime. This Service Level Agreement does not apply to any downtime, suspension, or termination of any HP services:

- that result in account suspension or termination due to breach of the Customer Agreement;

• caused by factors outside of our reasonable control, including any force majeure event or Internet access or related problems beyond the demarcation point of HP-controlled datacenters;

• that result from any actions or inactions of you or any third party; or

• that result from your equipment, software or other technology and/or third party equipment, software or other technology (other than those which are under our direct control).

The service credit remedy set forth in this Service Level Agreement is your sole and exclusive remedy for any failure to meet availability of HP Cloud Compute.

Table 9-6: Service Level Agreement from Amazon EC2

Effective Date: June 1, 2013:

This Amazon EC2 Service Level Agreement ("SLA") is a policy governing the use of Amazon Elastic Compute Cloud ("Amazon EC2") and Amazon Elastic Block Store ("Amazon EBS") under the terms of the Amazon Web Services Customer Agreement (the "AWS Agreement") between Amazon Web Services, Inc. ("AWS", "us" or "we") and users of AWS' services ("you"). This SLA applies separately to each account using Amazon EC2 or Amazon EBS. Unless otherwise provided herein, this SLA is subject to the terms of the AWS Agreement and capitalized terms will have the meaning specified in the AWS Agreement. We reserve the right to change the terms of this SLA in accordance with the AWS Agreement.

Service Commitment:

AWS will use commercially reasonable efforts to make Amazon EC2 and Amazon EBS each available with a Monthly Uptime Percentage (defined below) of at least 99.95%, in each case during any monthly billing cycle (the "Service Commitment"). In the event Amazon EC2 or Amazon EBS does not meet the Service Commitment, you will be eligible to receive a Service Credit as described below.

Definitions

• "Monthly Uptime Percentage" is calculated by subtracting from 100% the percentage of minutes during the month in which Amazon EC2 or Amazon EBS, as applicable, was in the state of "Region Unavailable." Monthly Uptime Percentage measurements exclude downtime resulting directly or indirectly from any Amazon EC2

SLA Exclusion (defined below).

• "Region Unavailable" and "Region Unavailability" mean that more than one Availability Zone in which you are running an instance, within the same Region, is "Unavailable" to you.

• "Unavailable" and "Unavailability" mean:

o For Amazon EC2, when all of your running instances have no external connectivity.

o For Amazon EBS, when all of your attached volumes perform zero read write IO, with pending IO in the queue.

• A "Service Credit" is a dollar credit, calculated as set forth below, that we may credit back to an eligible account.

Service Commitments and Service Credits

Service Credits are calculated as a percentage of the total charges paid by you (excluding one-time payments such as upfront payments made for Reserved Instances) for either Amazon EC2 or Amazon EBS (whichever was Unavailable, or both if both were Unavailable) in the Region affected for the monthly billing cycle in which the Region Unavailability occurred in accordance with the schedule below.

Monthly Uptime Percentage	Service Credit Percentage
Less than 99.95% but equal to or greater than 99.0%	10%
Less than 99.0%	30%

We will apply any Service Credits only against future Amazon EC2 or Amazon EBS payments otherwise due from you. At our discretion, we may issue the Service Credit to the credit card you used to pay for the billing cycle in which the Unavailability occurred. Service Credits will not entitle you to any refund or other payment from AWS. A Service Credit will be applicable and issued only if the credit amount for the applicable monthly billing cycle is greater than one dollar ($1 USD). Service Credits may not be transferred or applied to any other account. Unless otherwise provided in the AWS Agreement, your sole and exclusive remedy for any unavailability, non-performance, or other failure by us to provide Amazon EC2 or Amazon EBS is the receipt of a Service Credit (if eligible) in

accordance with the terms of this SLA.

Credit Request and Payment Procedures

To receive a Service Credit, you must submit a claim by opening a case in the AWS Support Center. To be eligible, the credit request must be received by us by the end of the second billing cycle after which the incident occurred and must include:

1. the words "SLA Credit Request" in the subject line;
2. the dates and times of each Unavailability incident that you are claiming;
3. the affected EC2 instance IDs or the affected EBS volume IDs; and
4. your request logs that document the errors and corroborate your claimed outage (any confidential or sensitive information in these logs should be removed or replaced with asterisks).

If the Monthly Uptime Percentage of such request is confirmed by us and is less than the Service Commitment, then we will issue the Service Credit to you within one billing cycle following the month in which your request is confirmed by us. Your failure to provide the request and other information as required above will disqualify you from receiving a Service Credit.

Amazon EC2 SLA Exclusions

The Service Commitment does not apply to any unavailability, suspension or termination of Amazon EC2 or Amazon EBS, or any other Amazon EC2 or Amazon EBS performance issues: (i) that result from a suspension described in Section 6.1 of the AWS Agreement; (ii) caused by factors outside of our reasonable control, including any force majeure event or Internet access or related problems beyond the demarcation point of Amazon EC2 or Amazon EBS; (iii) that result from any actions or inactions of you or any third party, including failure to acknowledge a recovery volume; (iv) that result from your equipment, software or other technology and/or third party equipment, software or other technology (other than third party equipment within our direct control); (v) that result from failures of individual instances or volumes not attributable to Region Unavailability; (vi) that result from any maintenance as provided for pursuant to the AWS Agreement; or (vii) arising from our suspension and termination of your right to use Amazon EC2 or Amazon EBS in accordance with the AWS Agreement (collectively, the "Amazon EC2 SLA Exclusions"). If availability is impacted by factors other than those used in our Monthly Uptime Percentage calculation, then we may issue a Service Credit considering such factors at our discretion.

10. Appendix-B

The mockup implementation of the Fuzzy inference System using steps from Chapter 4 based on two inputs and one output implemented in Microsoft Excel VBA programming is given in Table 10-1.

The mockup implementation of the Fuzzy Inference System using steps from Chapter 4 based on three inputs and one output implemented in Microsoft Excel VBA programming is given in Table 10-2.

Table 10-1: Mockup Implementation of FIS baed on two inputs and one output

```
Function FuzzyOnTwo(Input1 As Double, Input2 As Double) As Double
'This function is FIS Function based on Two Inputs and One Output
'Input Variables X1,X2 Declaration
Dim x1 As Double
Dim x2 As Double

'Declaration of Linguistic Variables
Dim LOW As Double
Dim MEDIUM As Double
Dim HIGH As Double

'Declaration of Rule Consequent Variables
Dim CONS1 As Double
Dim CONS2 As Double
Dim CONS3 As Double
Dim CONS4 As Double
Dim CONS5 As Double
Dim CONS6 As Double
Dim CONS7 As Double
Dim CONS8 As Double
Dim CONS9 As Double

'Declaration of Rule Strength Variables
Dim RS1 As Double
Dim RS2 As Double
Dim RS3 As Double
Dim RS4 As Double
Dim RS5 As Double
Dim RS6 As Double
Dim RS7 As Double
Dim RS8 As Double
```

```
Dim RS9 As Double

'Declaration of Variable to Check individual Rules Applied?
Dim Rule1Applied As Double
Dim Rule2Applied As Double
Dim Rule3Applied As Double
Dim Rule4Applied As Double
Dim Rule5Applied As Double
Dim Rule6Applied As Double
Dim Rule7Applied As Double
Dim Rule8Applied As Double
Dim Rule9Applied As Double

'Declaration of Rule Weight Variables
Dim W1 As Double
Dim W2 As Double
Dim W3 As Double
Dim W4 As Double
Dim W5 As Double
Dim W6 As Double
Dim W7 As Double
Dim W8 As Double
Dim W9 As Double

'Declaration of Variables used in Membership Functions
Dim I1Y1 As Double
Dim I1Y2 As Double
Dim I1Y3 As Double
Dim I1Y4 As Double
Dim I1Y5 As Double
Dim I1Y6 As Double

Dim I2Y1 As Double
Dim I2Y2 As Double
Dim I2Y3 As Double
Dim I2Y4 As Double
Dim I2Y5 As Double
Dim I2Y6 As Double

'Declaration of Variables to Check Each Input's MF Linguistic Variable
Dim ISX1LOW As Double
Dim ISX2LOW As Double

Dim ISX1MEDIUM As Double
Dim ISX2MEDIUM As Double

Dim ISX1HIGH As Double
Dim ISX2HIGH As Double
```

```
'Declaration of Rule Antecedent Variables
Dim ANT1 As Double
Dim ANT2 As Double
Dim ANT3 As Double
Dim ANT4 As Double
Dim ANT5 As Double
Dim ANT6 As Double
Dim ANT7 As Double
Dim ANT8 As Double
Dim ANT9 As Double

'Some Variable Initializations
LOW = 1
MEDIUM = 2
HIGH = 3

W1 = 1
W2 = 1
W3 = 1
W4 = 1
W5 = 1
W6 = 1
W7 = 1
W8 = 1
W9 = 1

I1Y1 = -1
I1Y2 = -1
I1Y3 = -1
I1Y4 = -1
I1Y5 = -1
I1Y6 = -1

I2Y1 = -1
I2Y2 = -1
I2Y3 = -1
I2Y4 = -1
I2Y5 = -1
I2Y6 = -1

RS1 = 0
RS2 = 0
RS3 = 0
RS4 = 0
RS5 = 0
RS6 = 0
RS7 = 0
```

```
RS8 = 0
RS9 = 0

CONS1 = 0
CONS2 = 0
CONS3 = 0
CONS4 = 0
CONS5 = 0
CONS6 = 0
CONS7 = 0
CONS8 = 0
CONS9 = 0

Rule1Applied = 0
Rule2Applied = 0
Rule3Applied = 0
Rule4Applied = 0
Rule5Applied = 0
Rule6Applied = 0
Rule7Applied = 0
Rule8Applied = 0
Rule9Applied = 0

x1 = Input1
x2 = Input2

'Start of FIS Process: Ehecking each MF Equestion for Input1 and Input2

   'Start with x1 Input equations

   If x1 = 0 Then
   I1Y1 = 1
   End If

   If x1 > 0 And x1 <= 4 Then
   I1Y2 = (-x1 / 4) + 1
   End If

   If x1 = 3 Then
   I1Y3 = 1
   End If

   If x1 > 3 And x1 <= 7 Then
   I1Y4 = ((-x1 + 3) / 4) + 1
   End If

   If x1 = 6 Then
   I1Y5 = 1
```

```
    End If

    If x1 > 6 And x1 <= 10 Then
    I1Y6 = ((-x1 + 6) / 4) + 1
    End If

    'Start with x2 Input equations

    If x2 = 0 Then
    I2Y1 = 1
    End If

    If x2 > 0 And x2 <= 4 Then
    I2Y2 = (-x2 / 4) + 1
    End If

    If x2 = 3 Then
    I2Y3 = 1
    End If

    If x2 > 3 And x2 <= 7 Then
    I2Y4 = ((-x2 + 3) / 4) + 1
    End If

    If x2 = 6 Then
    I2Y5 = 1
    End If

    If x2 > 6 And x2 <= 10 Then
    I2Y6 = ((-x2 + 6) / 4) + 1
    End If

' Fuzzy Associative Memory

' Rule 1: If Low and Low Then Low
' Rule 2: If Low and Medium Then Low
' Rule 3: If Low and High Then Medium
' Rule 4: If Medium and Low Then Low
' Rule 5: If Medium and Medium Then Medium
' Rule 6: If Medium and High Then Medium
' Rule 7: If High and Low Then Medium
' Rule 8: If High and Medium Then Medium
' Rule 9: If High and High Then High

" Check which Membership Functions Applied

ISX1LOW = MaxOfTwo(I1Y1, I1Y2)
ISX2LOW = MaxOfTwo(I2Y1, I2Y2)
```

```
ISX1MEDIUM = MaxOfTwo(I1Y3, I1Y4)
ISX2MEDIUM = MaxOfTwo(I2Y3, I2Y4)
ISX1HIGH = MaxOfTwo(I1Y5, I1Y6)
ISX2HIGH = MaxOfTwo(I2Y5, I2Y6)


'Rule 1: If Low and Low Then Low
If ISX1LOW <> -1 And ISX2LOW <> -1 Then
    ANT1 = MaxOfTwo(ISX1LOW, ISX2LOW)
    RS1 = ANT1 * W1
    CONS1 = FAMONTWO(LOW, LOW) + (1.665 - (1.665 * RS1))
    Rule1Applied = 1
End If

' Rule 2: If Low and Medium Then Low
If ISX1LOW <> -1 And ISX2MEDIUM <> -1 Then
    ANT2 = MaxOfTwo(ISX1LOW, ISX2MEDIUM)
    RS2 = ANT2 * W2
    CONS2 = FAMONTWO(LOW, MEDIUM) + (1.665 - (1.665 * RS2))
    Rule2Applied = 1
End If

' Rule 3: If Low and High Then Medium
If ISX1LOW <> -1 And ISX2HIGH <> -1 Then
    ANT3 = MaxOfTwo(ISX1LOW, ISX2HIGH)
    RS3 = ANT3 * W3
    CONS3 = FAMONTWO(LOW, HIGH) + (1.665 - (1.665 * RS3))
    Rule3Applied = 1
End If

' Rule 4: If Medium and Low Then Low
If ISX1MEDIUM <> -1 And ISX2LOW <> -1 Then
    ANT4 = MaxOfTwo(ISX1MEDIUM, ISX2LOW)
    RS4 = ANT4 * W4
    CONS4 = FAMONTWO(MEDIUM, LOW) + (1.665 - (1.665 * RS4))
    Rule4Applied = 1
End If

' Rule 5: If Medium and Medium Then Medium
If ISX1MEDIUM <> -1 And ISX2MEDIUM <> -1 Then
    ANT5 = MaxOfTwo(ISX1MEDIUM, ISX2MEDIUM)
    RS5 = ANT5 * W5
    CONS5 = FAMONTWO(MEDIUM, MEDIUM) + (1.665 - (1.665 * RS5))
    Rule5Applied = 1
End If

' Rule 6: If Medium and High Then Medium
If ISX1MEDIUM <> -1 And ISX2HIGH <> -1 Then
```

```
    ANT6 = MaxOfTwo(ISX1MEDIUM, ISX2HIGH)

    RS6 = ANT6 * W6
    CONS6 = FAMONTWO(MEDIUM, HIGH) + (1.665 - (1.665 * RS6))
    Rule6Applied = 1
End If

' Rule 7: If High and Low Then Medium
If ISX1HIGH <> -1 And ISX2LOW <> -1 Then
    ANT7 = MaxOfTwo(ISX1HIGH, ISX2LOW)
    RS7 = ANT7 * W7
    CONS7 = FAMONTWO(HIGH, LOW) + (1.665 - (1.665 * RS7))
    Rule7Applied = 1
End If

' Rule 8: If High and Medium Then Medium
If ISX1HIGH <> -1 And ISX2MEDIUM <> -1 Then
    ANT8 = MaxOfTwo(ISX1HIGH, ISX2MEDIUM)
    RS8 = ANT8 * W8
    CONS8 = FAMONTWO(HIGH, MEDIUM) + (1.665 - (1.665 * RS8))
    Rule8Applied = 1
End If

' Rule 9: If High and High Then High
If ISX1HIGH <> -1 And ISX2HIGH <> -1 Then
    ANT9 = MaxOfTwo(ISX1HIGH, ISX2HIGH)
    RS9 = ANT9 * W9
    CONS9 = FAMONTWO(HIGH, HIGH) + (1.665 - (1.665 * RS9))
    Rule9Applied = 1
End If

FuzzyOnTwo = (CONS1 + CONS2 + CONS3 + CONS4 + CONS5 + CONS6 + CONS7 + CONS8 + CONS9) / _
(Rule1Applied + Rule2Applied + Rule3Applied + Rule4Applied + Rule5Applied + Rule6Applied + Rule7Applied +
Rule8Applied + Rule9Applied)
End Function
```

```
Function MaxOfTwo(Input1 As Double, Input2 As Double) As Double
If Input1 > Input2 Then
MaxOfTwo = Input1
ElseIf Input2 > Input1 Then
MaxOfTwo = Input2
Else
MaxOfTwo = Input1
End If
End Function
```

```
Function FAMONTWO(Input1 As Double, Input2 As Double) As Double
Dim sum As Double
sum = (Input1 + Input2) / 2
```

```
'Weak Low
If sum >= 1 And sum < 1.5 Then
    FAMONTWO = 0

    'Strong Low
    ElseIf sum >= 1.5 And sum < 2 Then
    FAMONTWO = 1.665

    'Weak Medium
    ElseIf sum >= 2 And sum < 2.5 Then
    FAMONTWO = 3.33

    'Strong Medium
    ElseIf sum >= 2.5 And sum < 3 Then
    FAMONTWO = 4.995

    ElseIf sum = 3 Then

    'Choice Either Select Weak High or Strong High

    'Weak High
    'FAMONTWO = 6.665

    'Strong High = 8.325
    FAMONTWO = 8.325
End If
End Function
```

Table 10-2: Mockup Implementation of FIS based on Three Inputs and One Output

```
Function FuzzyOnThree(Input1 As Double, Input2 As Double, Input3 As Double) As Double
'This function is FIS Function based on Three Inputs and One Output
'Input Variables X1,X2 and X3 Declaration
Dim x1 As Double
Dim x2 As Double
Dim x3 As Double

'Declaration of Linguistic Variables
Dim LOW As Double
Dim MEDIUM As Double
Dim HIGH As Double

'Declaration of Rule Consequent Variables
Dim CONS1 As Double
```

```
Dim CONS2 As Double
Dim CONS3 As Double
Dim CONS4 As Double
Dim CONS5 As Double
Dim CONS6 As Double
Dim CONS7 As Double
Dim CONS8 As Double
Dim CONS9 As Double
Dim CONS10 As Double
Dim CONS11 As Double
Dim CONS12 As Double
Dim CONS13 As Double
Dim CONS14 As Double
Dim CONS15 As Double
Dim CONS16 As Double
Dim CONS17 As Double
Dim CONS18 As Double
Dim CONS19 As Double
Dim CONS20 As Double
Dim CONS21 As Double
Dim CONS22 As Double
Dim CONS23 As Double
Dim CONS24 As Double
Dim CONS25 As Double
Dim CONS26 As Double
Dim CONS27 As Double

'Declaration of Rule Strength Variables
Dim RS1 As Double
Dim RS2 As Double
Dim RS3 As Double
Dim RS4 As Double
Dim RS5 As Double
Dim RS6 As Double
Dim RS7 As Double
Dim RS8 As Double
Dim RS9 As Double
Dim RS10 As Double
Dim RS11 As Double
Dim RS12 As Double
Dim RS13 As Double
Dim RS14 As Double
Dim RS15 As Double
Dim RS16 As Double
Dim RS17 As Double
Dim RS18 As Double
Dim RS19 As Double
Dim RS20 As Double
```

```
Dim RS21 As Double
Dim RS22 As Double
Dim RS23 As Double
Dim RS24 As Double
Dim RS25 As Double
Dim RS26 As Double
Dim RS27 As Double

'Declaration of Variable to Check individual Rules Applied?
Dim Rule1Applied As Double
Dim Rule2Applied As Double
Dim Rule3Applied As Double
Dim Rule4Applied As Double
Dim Rule5Applied As Double
Dim Rule6Applied As Double
Dim Rule7Applied As Double
Dim Rule8Applied As Double
Dim Rule9Applied As Double
Dim Rule10Applied As Double
Dim Rule11Applied As Double
Dim Rule12Applied As Double
Dim Rule13Applied As Double
Dim Rule14Applied As Double
Dim Rule15Applied As Double
Dim Rule16Applied As Double
Dim Rule17Applied As Double
Dim Rule18Applied As Double
Dim Rule19Applied As Double
Dim Rule20Applied As Double
Dim Rule21Applied As Double
Dim Rule22Applied As Double
Dim Rule23Applied As Double
Dim Rule24Applied As Double
Dim Rule25Applied As Double
Dim Rule26Applied As Double
Dim Rule27Applied As Double

'Declaration of Rule Weight Variables
Dim W1 As Double
Dim W2 As Double
Dim W3 As Double
Dim W4 As Double
Dim W5 As Double
Dim W6 As Double
Dim W7 As Double
Dim W8 As Double
Dim W9 As Double
Dim W10 As Double
```

```
Dim W11 As Double
Dim W12 As Double
Dim W13 As Double
Dim W14 As Double
Dim W15 As Double
Dim W16 As Double
Dim W17 As Double
Dim W18 As Double
Dim W19 As Double
Dim W20 As Double
Dim W21 As Double
Dim W22 As Double
Dim W23 As Double
Dim W24 As Double
Dim W25 As Double
Dim W26 As Double
Dim W27 As Double

'Declaration of Variables used in Membership Functions
Dim I1Y1 As Double
Dim I1Y2 As Double
Dim I1Y3 As Double
Dim I1Y4 As Double
Dim I1Y5 As Double
Dim I1Y6 As Double

Dim I2Y1 As Double
Dim I2Y2 As Double
Dim I2Y3 As Double
Dim I2Y4 As Double
Dim I2Y5 As Double
Dim I2Y6 As Double

Dim I3Y1 As Double
Dim I3Y2 As Double
Dim I3Y3 As Double
Dim I3Y4 As Double
Dim I3Y5 As Double
Dim I3Y6 As Double

'Declaration of Variables to Check Each Input's MF Linguistic Variable
Dim ISX1LOW As Double
Dim ISX2LOW As Double
Dim ISX3LOW As Double
Dim ISX1MEDIUM As Double
Dim ISX2MEDIUM As Double
Dim ISX3MEDIUM As Double
Dim ISX1HIGH As Double
```

```
Dim ISX2HIGH As Double
Dim ISX3HIGH As Double

'Declaration of Rule Antecedent Variables
Dim ANT1 As Double
Dim ANT2 As Double
Dim ANT3 As Double
Dim ANT4 As Double
Dim ANT5 As Double
Dim ANT6 As Double
Dim ANT7 As Double
Dim ANT8 As Double
Dim ANT9 As Double
Dim ANT10 As Double
Dim ANT11 As Double
Dim ANT12 As Double
Dim ANT13 As Double
Dim ANT14 As Double
Dim ANT15 As Double
Dim ANT16 As Double
Dim ANT17 As Double
Dim ANT18 As Double
Dim ANT19 As Double
Dim ANT20 As Double
Dim ANT21 As Double
Dim ANT22 As Double
Dim ANT23 As Double
Dim ANT24 As Double
Dim ANT25 As Double
Dim ANT26 As Double
Dim ANT27 As Double

'Some Initializations

LOW = 1
MEDIUM = 2
HIGH = 3

W1 = 1
W2 = 1
W3 = 1
W4 = 1
W5 = 1
W6 = 1
W7 = 1
W8 = 1
W9 = 1
W10 = 1
```

W11 = 1
W12 = 1
W13 = 1
W14 = 1
W15 = 1
W16 = 1
W17 = 1
W18 = 1
W19 = 1
W20 = 1
W21 = 1
W22 = 1
W23 = 1
W24 = 1
W25 = 1
W26 = 1
W27 = 1

I1Y1 = -1
I1Y2 = -1
I1Y3 = -1
I1Y4 = -1
I1Y5 = -1
I1Y6 = -1

I2Y1 = -1
I2Y2 = -1
I2Y3 = -1
I2Y4 = -1
I2Y5 = -1
I2Y6 = -1

I3Y1 = -1
I3Y2 = -1
I3Y3 = -1
I3Y4 = -1
I3Y5 = -1
I3Y6 = -1

RS1 = 0
RS2 = 0
RS3 = 0
RS4 = 0
RS5 = 0
RS6 = 0
RS7 = 0
RS8 = 0
RS9 = 0

RS10 = 0
RS11 = 0
RS12 = 0
RS13 = 0
RS14 = 0
RS15 = 0
RS16 = 0
RS17 = 0
RS18 = 0
RS19 = 0
RS20 = 0
RS21 = 0
RS22 = 0
RS23 = 0
RS24 = 0
RS25 = 0
RS26 = 0
RS27 = 0

CONS1 = 0
CONS2 = 0
CONS3 = 0
CONS4 = 0
CONS5 = 0
CONS6 = 0
CONS7 = 0
CONS8 = 0
CONS9 = 0
CONS10 = 0
CONS11 = 0
CONS12 = 0
CONS13 = 0
CONS14 = 0
CONS15 = 0
CONS16 = 0
CONS17 = 0
CONS18 = 0
CONS19 = 0
CONS20 = 0
CONS21 = 0
CONS22 = 0
CONS23 = 0
CONS24 = 0
CONS25 = 0
CONS26 = 0
CONS27 = 0

```
Rule1Applied = 0
Rule2Applied = 0
Rule3Applied = 0
Rule4Applied = 0
Rule5Applied = 0
Rule6Applied = 0
Rule7Applied = 0
Rule8Applied = 0
Rule9Applied = 0
Rule10Applied = 0
Rule11Applied = 0
Rule12Applied = 0
Rule13Applied = 0
Rule14Applied = 0
Rule15Applied = 0
Rule16Applied = 0
Rule17Applied = 0
Rule18Applied = 0
Rule19Applied = 0
Rule20Applied = 0
Rule21Applied = 0
Rule22Applied = 0
Rule23Applied = 0
Rule24Applied = 0
Rule25Applied = 0
Rule26Applied = 0
Rule27Applied = 0

x1 = Input1
x2 = Input2
x3 = Input3

'Start of FIS Process: Ehecking each MF Equestion for Input1, Input2 and Input3

   'Start with x1 Input equations
   If x1 = 0 Then
   I1Y1 = 1
   End If

   If x1 > 0 And x1 <= 4 Then
   I1Y2 = (-x1 / 4) + 1
   End If

   If x1 = 3 Then
   I1Y3 = 1
   End If

   If x1 > 3 And x1 <= 7 Then
```

```
I1Y4 = ((-x1 + 3) / 4) + 1
End If

If x1 = 6 Then
I1Y5 = 1
End If

If x1 > 6 And x1 <= 10 Then
I1Y6 = ((-x1 + 6) / 4) + 1
End If

'Start with x2 Input equations
If x2 = 0 Then
I2Y1 = 1
End If

If x2 > 0 And x2 <= 4 Then
I2Y2 = (-x2 / 4) + 1
End If

If x2 = 3 Then
I2Y3 = 1
End If

If x2 > 3 And x2 <= 7 Then
I2Y4 = ((-x2 + 3) / 4) + 1
End If

If x2 = 6 Then
I2Y5 = 1
End If

If x2 > 6 And x2 <= 10 Then
I2Y6 = ((-x2 + 6) / 4) + 1
End If


'Start with x3 Input equations
If x3 = 0 Then
I3Y1 = 1
End If

If x3 > 0 And x3 <= 4 Then
I3Y2 = (-x3 / 4) + 1
End If

If x3 = 3 Then
I3Y3 = 1
```

```
    End If

    If x3 > 3 And x3 <= 7 Then
    I3Y4 = ((-x3 + 3) / 4) + 1
    End If

    If x3 = 6 Then
    I3Y5 = 1
    End If

    If x3 > 6 And x3 <= 10 Then
    I3Y6 = ((-x3 + 6) / 4) + 1
    End If

' Fuzzy Associative Memory
' Rule 1: If Low and Low and Low Then Low(Weak Low)
' Rule 2: If Low and Low and Medium Then Low(Strong Low)
' Rule 3: If  Low and Low and High Then Medium(Weak Medium)
' Rule 4: If  Low and Medium and Low Then Low(Strong Low)
' Rule 5: If Low and Medium and Medium Then Medium(Weak Medium)
' Rule 6: If Low and Medium and High Then Medium(Strong Medium)
' Rule 7: If  Low and High  and Low Then Medium(Weak Medium)
' Rule 8: If  Low and High  and Medium Then Medium(Strong Medium)
' Rule 9: If  Low and High and High Then High(Weak High)
' Rule 10: If  Medium and Low  and Low Then Low(Strong Low)
' Rule 11: If  Medium and Low and Medium Then Medium(Weak Medium)
' Rule 12: If  Medium and Low and High Then Medium(Strong Medium)
' Rule 13: If  Medium and Medium  and Low Then Medium(Weak Medium)
' Rule 14: If  Medium and Medium and Medium Then Medium(Strong Medium)
' Rule 15: If  Medium and Medium and High Then High(Weak High)
' Rule 16: If  Medium and High and Low Then Medium(Strong Medium)
' Rule 17: If  Medium and High and Medium Then High(Weak High)
' Rule 18: If  Medium and High  and High Then High(Strong High)
' Rule 19: If  High and Low and Low Then Medium(Weak Medium)
' Rule 20: If  High and Low and Medium Then Medium(Strong Medium)
' Rule 21: If  High and Low and High Then High(Weak High)
' Rule 22: If  High and Medium and Low Then Medium(Strong Medium)
' Rule 23: If  High and Medium and Medium Then High(Weak High)
' Rule 24: If  High and Medium and High Then High(Strong High)
' Rule 25: If  High and High  and Low Then High(Weak High)
' Rule 26: If  High and High  and Medium Then High(Strong High)
' Rule 27: If  High and High and High Then High(Strong High)

'MF: Low   : Y1 , Y2
'MF: Medium: Y3 , Y4
'MF: High  : Y5 , Y6

ISX1LOW = MaxOfTwo(I1Y1, I1Y2)
```

```
ISX2LOW = MaxOfTwo(I2Y1, I2Y2)
ISX3LOW = MaxOfTwo(I3Y1, I3Y2)

ISX1MEDIUM = MaxOfTwo(I1Y3, I1Y4)
ISX2MEDIUM = MaxOfTwo(I2Y3, I2Y4)
ISX3MEDIUM = MaxOfTwo(I3Y3, I3Y4)

ISX1HIGH = MaxOfTwo(I1Y5, I1Y6)
ISX2HIGH = MaxOfTwo(I2Y5, I2Y6)
ISX3HIGH = MaxOfTwo(I3Y5, I3Y6)

'Rule 1: If Low and Low and Low Then Low(Weak Low)
   If ISX1LOW <> -1 And ISX2LOW <> -1 And ISX3LOW <> -1 Then
   ANT1 = MaxOfThree(ISX1LOW, ISX2LOW, ISX3LOW)
   RS1 = ANT1 * W1
   CONS1 = FAMONTHREE(LOW, LOW, LOW) + (1.665 - (1.665 * RS1))
   Rule1Applied = 1
   End If

' Rule 2: If Low and Low and Medium Then Low(Strong Low)
   If ISX1LOW <> -1 And ISX2LOW <> -1 And ISX3MEDIUM <> -1 Then
   ANT2 = MaxOfThree(ISX1LOW, ISX2LOW, ISX3MEDIUM)
   RS2 = ANT2 * W2
   CONS2 = FAMONTHREE(LOW, LOW, MEDIUM) + (1.665 - (1.665 * RS2))
   Rule2Applied = 1
   End If

' Rule 3: If  Low and Low and High Then Medium(Weak Medium)
   If ISX1LOW <> -1 And ISX2LOW <> -1 And ISX3HIGH <> -1 Then
   ANT3 = MaxOfThree(ISX1LOW, ISX2LOW, ISX3HIGH)
   RS3 = ANT3 * W3
   CONS3 = FAMONTHREE(LOW, LOW, HIGH) + (1.665 - (1.665 * RS3))
   Rule3Applied = 1
   End If

' Rule 4: If  Low and Medium and Low Then Low(Strong Low)
   If ISX1LOW <> -1 And ISX2MEDIUM <> -1 And ISX3LOW <> -1 Then
   ANT4 = MaxOfThree(ISX1LOW, ISX2MEDIUM, ISX3LOW)
   RS4 = ANT4 * W4
   CONS4 = FAMONTHREE(LOW, MEDIUM, LOW) + (1.665 - (1.665 * RS4))
   Rule4Applied = 1
    End If

' Rule 5: If Low and Medium and Medium Then Medium(Weak Medium)
   If ISX1LOW <> -1 And ISX2MEDIUM <> -1 And ISX3MEDIUM <> -1 Then
   ANT5 = MaxOfThree(ISX1LOW, ISX2MEDIUM, ISX3MEDIUM)
   RS5 = ANT5 * W5
   CONS5 = FAMONTHREE(LOW, MEDIUM, MEDIUM) + (1.665 - (1.665 * RS5))
```

```
    Rule5Applied = 1
    End If

' Rule 6: If Low and Medium and High Then Medium(Strong Medium)
    If ISX1LOW <> -1 And ISX2MEDIUM <> -1 And ISX3HIGH <> -1 Then
    ANT6 = MaxOfThree(ISX1LOW, ISX2MEDIUM, ISX3HIGH)
    RS6 = ANT6 * W6
    CONS6 = FAMONTHREE(LOW, MEDIUM, HIGH) + (1.665 - (1.665 * RS6))
    Rule6Applied = 1
    End If

' Rule 7: If  Low and High  and Low Then Medium(Weak Medium)
    If ISX1LOW <> -1 And ISX2HIGH <> -1 And ISX3LOW <> -1 Then
    ANT7 = MaxOfThree(ISX1LOW, ISX2HIGH, ISX3LOW)
    RS7 = ANT7 * W7
    CONS7 = FAMONTHREE(LOW, HIGH, LOW) + (1.665 - (1.665 * RS7))
    Rule7Applied = 1
    End If

' Rule 8: If  Low and High  and Medium Then Medium(Strong Medium)
    If ISX1LOW <> -1 And ISX2HIGH <> -1 And ISX3MEDIUM <> -1 Then
    ANT8 = MaxOfThree(ISX1LOW, ISX2HIGH, ISX3MEDIUM)
    RS8 = ANT8 * W8
    CONS8 = FAMONTHREE(LOW, HIGH, MEDIUM) + (1.665 - (1.665 * RS8))
    Rule8Applied = 1
    End If

' Rule 9: If  Low and High and High Then High(Weak High)
    If ISX1LOW <> -1 And ISX2HIGH <> -1 And ISX3HIGH <> -1 Then
    ANT9 = MaxOfThree(ISX1LOW, ISX2HIGH, ISX3HIGH)
    RS9 = ANT9 * W9
    CONS9 = FAMONTHREE(LOW, HIGH, HIGH) + (1.665 - (1.665 * RS9))
    Rule9Applied = 1
    End If

' Rule 10: If  Medium and Low  and Low Then Low(Strong Low)
    If ISX1MEDIUM <> -1 And ISX2LOW <> -1 And ISX3LOW <> -1 Then
    ANT10 = MaxOfThree(ISX1MEDIUM, ISX2LOW, ISX3LOW)
    RS10 = ANT10 * W10
    CONS10 = FAMONTHREE(MEDIUM, LOW, LOW) + (1.665 - (1.665 * RS10))
    Rule10Applied = 1
    End If

' Rule 11: If  Medium and Low and Medium Then Medium(Weak Medium)
    If ISX1MEDIUM <> -1 And ISX2LOW <> -1 And ISX3MEDIUM <> -1 Then
    ANT11 = MaxOfThree(ISX1MEDIUM, ISX2LOW, ISX3MEDIUM)
    RS11 = ANT11 * W11
    CONS11 = FAMONTHREE(MEDIUM, LOW, MEDIUM) + (1.665 - (1.665 * RS11))
```

```
    Rule11Applied = 1
    End If

' Rule 12: If  Medium and Low and High Then Medium(Strong Medium)
    If ISX1MEDIUM <> -1 And ISX2LOW <> -1 And ISX3HIGH <> -1 Then
    ANT12 = MaxOfThree(ISX1MEDIUM, ISX2LOW, ISX3HIGH)
    RS12 = ANT12 * W12
    CONS12 = FAMONTHREE(MEDIUM, LOW, HIGH) + (1.665 - (1.665 * RS12))
    Rule12Applied = 1
    End If

' Rule 13: If  Medium and Medium  and Low Then Medium(Weak Medium)
    If ISX1MEDIUM <> -1 And ISX2MEDIUM <> -1 And ISX3LOW <> -1 Then
    ANT13 = MaxOfThree(ISX1MEDIUM, ISX2MEDIUM, ISX3LOW)
    RS13 = ANT13 * W13
    CONS13 = FAMONTHREE(MEDIUM, MEDIUM, LOW) + (1.665 - (1.665 * RS13))
    Rule13Applied = 1
    End If

' Rule 14: If  Medium and Medium and Medium Then Medium(Strong Medium)
    If ISX1MEDIUM <> -1 And ISX2MEDIUM <> -1 And ISX3MEDIUM <> -1 Then
    ANT14 = MaxOfThree(ISX1MEDIUM, ISX2MEDIUM, ISX3MEDIUM)
    RS14 = ANT14 * W14
    CONS14 = FAMONTHREE(MEDIUM, MEDIUM, MEDIUM) + (1.665 - (1.665 * RS14))
    Rule14Applied = 1
    End If

' Rule 15: If  Medium and Medium and High Then High(Weak High)
    If ISX1MEDIUM <> -1 And ISX2MEDIUM <> -1 And ISX3HIGH <> -1 Then
    ANT15 = MaxOfThree(ISX1MEDIUM, ISX2MEDIUM, ISX3HIGH)
    RS15 = ANT15 * W15
    CONS15 = FAMONTHREE(MEDIUM, MEDIUM, HIGH) + (1.665 - (1.665 * RS15))
    Rule15Applied = 1
    End If

' Rule 16: If  Medium and High and Low Then Medium(Strong Medium)
    If ISX1MEDIUM <> -1 And ISX2HIGH <> -1 And ISX3LOW <> -1 Then
    ANT16 = MaxOfThree(ISX1MEDIUM, ISX2HIGH, ISX3LOW)
    RS16 = ANT16 * W16
    CONS16 = FAMONTHREE(MEDIUM, HIGH, LOW) + (1.665 - (1.665 * RS16))
    Rule16Applied = 1
    End If

' Rule 17: If  Medium and High and Medium Then High(Weak High)
    If ISX1MEDIUM <> -1 And ISX2HIGH <> -1 And ISX3MEDIUM <> -1 Then
    ANT17 = MaxOfThree(ISX1MEDIUM, ISX2HIGH, ISX3MEDIUM)
    RS17 = ANT17 * W17
    CONS17 = FAMONTHREE(MEDIUM, HIGH, MEDIUM) + (1.665 - (1.665 * RS17))
```

```
    Rule17Applied = 1

    End If

' Rule 18: If  Medium and High  and High Then High(Strong High)
    If ISX1MEDIUM <> -1 And ISX2HIGH <> -1 And ISX3HIGH <> -1 Then
    ANT18 = MaxOfThree(ISX1MEDIUM, ISX2HIGH, ISX3HIGH)
    RS18 = ANT18 * W18
    CONS18 = FAMONTHREE(MEDIUM, HIGH, HIGH) + (1.665 - (1.665 * RS18))
    Rule18Applied = 1
    End If

' Rule 19: If  High and Low and Low Then Medium(Weak Medium)
    If ISX1HIGH <> -1 And ISX2LOW <> -1 And ISX3LOW <> -1 Then
    ANT19 = MaxOfThree(ISX1HIGH, ISX2LOW, ISX3LOW)
    RS19 = ANT19 * W19
    CONS19 = FAMONTHREE(HIGH, LOW, LOW) + (1.665 - (1.665 * RS19))
    Rule19Applied = 1
    End If

' Rule 20: If  High and Low and Medium Then Medium(Strong Medium)
    If ISX1HIGH <> -1 And ISX2LOW <> -1 And ISX3MEDIUM <> -1 Then
    ANT20 = MaxOfThree(ISX1HIGH, ISX2LOW, ISX3MEDIUM)
    RS20 = ANT20 * W20
    CONS20 = FAMONTHREE(HIGH, LOW, MEDIUM) + (1.665 - (1.665 * RS20))
    Rule20Applied = 1
    End If

' Rule 21: If  High and Low and High Then High(Weak High)
    If ISX1HIGH <> -1 And ISX2LOW <> -1 And ISX3HIGH <> -1 Then
    ANT21 = MaxOfThree(ISX1HIGH, ISX2LOW, ISX3HIGH)
    RS21 = ANT21 * W21
    CONS21 = FAMONTHREE(HIGH, LOW, HIGH) + (1.665 - (1.665 * RS21))
    Rule21Applied = 1
    End If

' Rule 22: If  High and Medium and Low Then Medium(Strong Medium)
    If ISX1HIGH <> -1 And ISX2MEDIUM <> -1 And ISX3LOW <> -1 Then
    ANT22 = MaxOfThree(ISX1HIGH, ISX2MEDIUM, ISX3LOW)
    RS22 = ANT22 * W22
    CONS22 = FAMONTHREE(HIGH, MEDIUM, LOW) + (1.665 - (1.665 * RS22))
    Rule22Applied = 1
    End If

' Rule 23: If  High and Medium and Medium Then High(Weak High)
    If ISX1HIGH <> -1 And ISX2MEDIUM <> -1 And ISX3MEDIUM <> -1 Then
    ANT23 = MaxOfThree(ISX1HIGH, ISX2MEDIUM, ISX3MEDIUM)
    RS23 = ANT23 * W23
```

```
        CONS23 = FAMONTHREE(HIGH, MEDIUM, MEDIUM) + (1.665 - (1.665 * RS23))
        Rule23Applied = 1
        End If

    ' Rule 24: If  High and Medium and High Then High(Strong High)
        If ISX1HIGH <> -1 And ISX2MEDIUM <> -1 And ISX3HIGH <> -1 Then
        ANT24 = MaxOfThree(ISX1HIGH, ISX2MEDIUM, ISX3HIGH)
        RS24 = ANT24 * W24
       CONS24 = FAMONTHREE(HIGH, MEDIUM, HIGH) + (1.665 - (1.665 * RS24))
        Rule24Applied = 1
        End If

    ' Rule 25: If  High and High  and Low Then High(Weak High)
        If ISX1HIGH <> -1 And ISX2HIGH <> -1 And ISX3LOW <> -1 Then
        ANT25 = MaxOfThree(ISX1HIGH, ISX2HIGH, ISX3LOW)
        RS25 = ANT25 * W25
       CONS25 = FAMONTHREE(HIGH, HIGH, LOW) + (1.665 - (1.665 * RS25))
        Rule25Applied = 1
        End If

    ' Rule 26: If  High and High  and Medium Then High(Strong High)
        If ISX1HIGH <> -1 And ISX2HIGH <> -1 And ISX3MEDIUM <> -1 Then
        ANT26 = MaxOfThree(ISX1HIGH, ISX2HIGH, ISX3MEDIUM)
        RS26 = ANT26 * W26
        CONS26 = FAMONTHREE(HIGH, HIGH, MEDIUM) + (1.665 - (1.665 * RS26))
        Rule26Applied = 1
        End If

    ' Rule 27: If  High and High and High Then High(Strong High)
       If ISX1HIGH <> -1 And ISX2HIGH <> -1 And ISX3HIGH <> -1 Then
        ANT27 = MaxOfThree(ISX1HIGH, ISX2HIGH, ISX3HIGH)
        RS27 = ANT27 * W27
        CONS27 = FAMONTHREE(HIGH, HIGH, HIGH) + (1.665 - (1.665 * RS27))
        Rule27Applied = 1
        End If

    FuzzyOnThree = (CONS1 + CONS2 + CONS3 + CONS4 + CONS5 + CONS6 + CONS7 + CONS8 + CONS9 + CONS10 +
CONS11 + CONS12 + CONS13 + CONS14 _
    + CONS15 + CONS16 + CONS17 + CONS18 + CONS19 + CONS20 + CONS21 + CONS22 + CONS23 + CONS24 +
CONS25 + CONS26 + CONS27) / _
    (Rule1Applied + Rule2Applied + Rule3Applied + Rule4Applied + Rule5Applied + Rule6Applied + Rule7Applied +
Rule8Applied + Rule9Applied _
    + Rule10Applied + Rule11Applied + Rule12Applied + Rule13Applied + Rule14Applied + Rule15Applied + Rule16Applied
+ Rule17Applied + Rule18Applied _
    + Rule19Applied + Rule20Applied + Rule21Applied + Rule22Applied + Rule23Applied + Rule24Applied + Rule25Applied
+ Rule26Applied + Rule27Applied)
```

End Function

```
Function MaxOfThree(Input1 As Double, Input2 As Double, Input3 As Double) As Double
Dim tempmax As Double
tempmax = -1

If Input1 = -1 And Input2 = -1 And Input3 = -1 Then
MaxOfThree = -1
GoTo endoffun
End If

If Input1 > tempmax Then
tempmax = Input1
End If

If Input2 > tempmax Then
tempmax = Input2
End If

If Input3 > tempmax Then
tempmax = Input3
End If

MaxOfThree = tempmax
endoffun:
End Function
```

```
Function FAMONTHREE(Input1 As Double, Input2 As Double, Input3 As Double) As Double

Dim sum As Double

sum = (Input1 + Input2 + Input3) / 3

'Weak Low
If sum >= 1 And sum < 1.33 Then
FAMONTHREE = 0

'Strong Low
ElseIf sum >= 1.33 And sum < 2 Then
FAMONTHREE = 1.665

'Weak Medium
ElseIf sum >= 2 And sum < 2.33 Then
FAMONTHREE = 3.33

'Strong Medium
ElseIf sum >= 2.33 And sum < 2.66 Then
FAMONTHREE = 4.995

'Weak High
ElseIf sum >= 2.66 And sum < 3 Then
```

```
FAMONTHREE = 6.665

'Strong High
ElseIf sum = 3 Then
'FAMONTHREE = 8.325
FAMONTHREE = 8.325
'FAMONTHREE = 6.665
End If
End Function
```

Table 10-3: FIS Mockup calculations based on Three Inputs and One Output

S.No	I1	I2	I3	FIS Output	S.No	I1	I2	I3	FIS Output	S.No	I1	I2	I3	FIS Output
1	10	10	0	4.995	445	7	3	4	3.0178125	889	3	7	8	5.3084375
2	10	10	1	5.41125	446	7	3	5	3.6421875	890	3	7	9	5.4125
3	10	10	2	5.8275	447	7	3	6	4.37125	891	3	7	10	5.4125
4	10	10	3	6.454375	448	7	3	7	4.4753125	892	3	6	0	2.08125
5	10	10	4	6.870625	449	7	3	8	5.3084375	893	3	6	1	2.1853125
6	10	10	5	7.4975	450	7	3	9	5.4125	894	3	6	2	2.289375
7	10	10	6	8.119375	451	7	3	10	5.4125	895	3	6	3	2.86171875
8	10	10	7	8.535625	452	7	2	0	1.665	896	3	6	4	2.91375
9	10	10	8	9.1575	453	7	2	1	2.08125	897	3	6	5	3.538125
10	10	10	9	9.57375	454	7	2	2	2.289375	898	3	6	6	4.31921875
11	10	10	10	9.99	455	7	2	3	2.3934375	899	3	6	7	4.37125
12	10	9	0	4.995	456	7	2	4	2.6015625	900	3	6	8	5.204375
13	10	9	1	5.41125	457	7	2	5	3.121875	901	3	6	9	5.3084375
14	10	9	2	5.8275	458	7	2	6	3.6421875	902	3	6	10	5.3084375
15	10	9	3	6.454375	459	7	2	7	3.8503125	903	3	5	0	1.665
16	10	9	4	6.6625	460	7	2	8	4.786875	904	3	5	1	1.873125
17	10	9	5	7.4975	461	7	2	9	4.786875	905	3	5	2	2.08125
18	10	9	6	8.119375	462	7	2	10	4.786875	906	3	5	3	2.289375
19	10	9	7	8.3275	463	7	1	0	1.665	907	3	5	4	2.3934375
20	10	9	8	9.1575	464	7	1	1	2.08125	908	3	5	5	2.91375
21	10	9	9	9.57375	465	7	1	2	2.08125	909	3	5	6	3.538125
22	10	9	10	9.57375	466	7	1	3	2.289375	910	3	5	7	3.6421875
23	10	8	0	4.995	467	7	1	4	2.4975	911	3	5	8	4.57875
24	10	8	1	5.41125	468	7	1	5	2.91375	912	3	5	9	4.57875
25	10	8	2	5.8275	469	7	1	6	3.538125	913	3	5	10	4.57875
26	10	8	3	6.24625	470	7	1	7	3.74625	914	3	4	0	1.24875
27	10	8	4	6.454375	471	7	1	8	4.57875	915	3	4	1	1.456875
28	10	8	5	7.4975	472	7	1	9	4.57875	916	3	4	2	1.5609375
29	10	8	6	7.91125	473	7	1	10	4.57875	917	3	4	3	1.873125
30	10	8	7	8.119375	474	7	0	0	1.665	918	3	4	4	1.9771875

31	10	8	8	9.1575	475	7	0	1	1.665	919	3	4	5	2.3934375
32	10	8	9	9.1575	476	7	0	2	1.665	920	3	4	6	2.91375
33	10	8	10	9.1575	477	7	0	3	2.08125	921	3	4	7	3.0178125
34	10	7	0	4.1625	478	7	0	4	2.08125	922	3	4	8	3.6421875
35	10	7	1	4.57875	479	7	0	5	2.4975	923	3	4	9	3.74625
36	10	7	2	4.786875	480	7	0	6	3.33	924	3	4	10	3.74625
37	10	7	3	5.4125	481	7	0	7	3.33	925	3	3	0	1.24875
38	10	7	4	5.7246875	482	7	0	8	4.1625	926	3	3	1	1.3528125
39	10	7	5	6.454375	483	7	0	9	4.1625	927	3	3	2	1.456875
40	10	7	6	7.07875	484	7	0	10	4.1625	928	3	3	3	1.82109375
41	10	7	7	7.3909375	485	6	10	0	4.1625	929	3	3	4	1.873125
42	10	7	8	8.119375	486	6	10	1	4.370625	930	3	3	5	2.289375
43	10	7	9	8.3275	487	6	10	2	4.57875	931	3	3	6	2.86171875
44	10	7	10	8.535625	488	6	10	3	5.3084375	932	3	3	7	2.91375
45	10	6	0	4.1625	489	6	10	4	5.4125	933	3	3	8	3.538125
46	10	6	1	4.370625	490	6	10	5	6.24625	934	3	3	9	3.6421875
47	10	6	2	4.57875	491	6	10	6	6.9746875	935	3	3	10	3.6421875
48	10	6	3	5.3084375	492	6	10	7	7.07875	936	3	2	0	0.8325
49	10	6	4	5.4125	493	6	10	8	7.91125	937	3	2	1	1.040625
50	10	6	5	6.24625	494	6	10	9	8.119375	938	3	2	2	1.24875
51	10	6	6	6.9746875	495	6	10	10	8.119375	939	3	2	3	1.456875
52	10	6	7	7.07875	496	6	9	0	4.1625	940	3	2	4	1.5609375
53	10	6	8	7.91125	497	6	9	1	4.370625	941	3	2	5	2.08125
54	10	6	9	8.119375	498	6	9	2	4.57875	942	3	2	6	2.289375
55	10	6	10	8.119375	499	6	9	3	5.3084375	943	3	2	7	2.3934375
56	10	5	0	3.33	500	6	9	4	5.4125	944	3	2	8	2.91375
57	10	5	1	3.74625	501	6	9	5	6.24625	945	3	2	9	2.91375
58	10	5	2	4.1625	502	6	9	6	6.9746875	946	3	2	10	2.91375
59	10	5	3	4.57875	503	6	9	7	7.07875	947	3	1	0	0.8325
60	10	5	4	4.786875	504	6	9	8	7.91125	948	3	1	1	1.040625
61	10	5	5	5.8275	505	6	9	9	8.119375	949	3	1	2	1.040625
62	10	5	6	6.24625	506	6	9	10	8.119375	950	3	1	3	1.3528125
63	10	5	7	6.454375	507	6	8	0	4.1625	951	3	1	4	1.456875
64	10	5	8	7.4975	508	6	8	1	4.370625	952	3	1	5	1.873125
65	10	5	9	7.4975	509	6	8	2	4.57875	953	3	1	6	2.1853125
66	10	5	10	7.4975	510	6	8	3	5.204375	954	3	1	7	2.289375
67	10	4	0	2.4975	511	6	8	4	5.3084375	955	3	1	8	2.705625
68	10	4	1	2.91375	512	6	8	5	6.24625	956	3	1	9	2.705625
69	10	4	2	3.121875	513	6	8	6	6.870625	957	3	1	10	2.705625
70	10	4	3	3.74625	514	6	8	7	6.9746875	958	3	0	0	0.8325
71	10	4	4	4.0584375	515	6	8	8	7.91125	959	3	0	1	0.8325
72	10	4	5	4.786875	516	6	8	9	7.91125	960	3	0	2	0.8325
73	10	4	6	5.4125	517	6	8	10	7.91125	961	3	0	3	1.24875

74	10	4	7	5.7246875	518	6	7	0	3.33	962	3	0	4	1.24875
75	10	4	8	6.454375	519	6	7	1	3.538125	963	3	0	5	1.665
76	10	4	9	6.6625	520	6	7	2	3.6421875	964	3	0	6	2.08125
77	10	4	10	6.870625	521	6	7	3	4.37125	965	3	0	7	2.08125
78	10	3	0	2.4975	522	6	7	4	4.4753125	966	3	0	8	2.4975
79	10	3	1	2.705625	523	6	7	5	5.3084375	967	3	0	9	2.4975
80	10	3	2	2.91375	524	6	7	6	6.0375	968	3	0	10	2.4975
81	10	3	3	3.6421875	525	6	7	7	6.1415625	969	2	10	0	1.665
82	10	3	4	3.74625	526	6	7	8	6.9746875	970	2	10	1	2.08125
83	10	3	5	4.57875	527	6	7	9	7.07875	971	2	10	2	2.4975
84	10	3	6	5.3084375	528	6	7	10	7.07875	972	2	10	3	2.91375
85	10	3	7	5.4125	529	6	6	0	3.33	973	2	10	4	3.121875
86	10	3	8	6.24625	530	6	6	1	3.4340625	974	2	10	5	4.1625
87	10	3	9	6.454375	531	6	6	2	3.538125	975	2	10	6	4.57875
88	10	3	10	6.454375	532	6	6	3	4.31921875	976	2	10	7	4.786875
89	10	2	0	1.665	533	6	6	4	4.37125	977	2	10	8	5.8275
90	10	2	1	2.08125	534	6	6	5	5.204375	978	2	10	9	5.8275
91	10	2	2	2.4975	535	6	6	6	5.98546875	979	2	10	10	5.8275
92	10	2	3	2.91375	536	6	6	7	6.0375	980	2	9	0	1.665
93	10	2	4	3.121875	537	6	6	8	6.870625	981	2	9	1	2.08125
94	10	2	5	4.1625	538	6	6	9	6.9746875	982	2	9	2	2.4975
95	10	2	6	4.57875	539	6	6	10	6.9746875	983	2	9	3	2.91375
96	10	2	7	4.786875	540	6	5	0	2.4975	984	2	9	4	3.121875
97	10	2	8	5.8275	541	6	5	1	2.705625	985	2	9	5	4.1625
98	10	2	9	5.8275	542	6	5	2	2.91375	986	2	9	6	4.57875
99	10	2	10	5.8275	543	6	5	3	3.538125	987	2	9	7	4.786875
100	10	1	0	1.665	544	6	5	4	3.6421875	988	2	9	8	5.8275
101	10	1	1	2.08125	545	6	5	5	4.57875	989	2	9	9	5.8275
102	10	1	2	2.08125	546	6	5	6	5.204375	990	2	9	10	5.8275
103	10	1	3	2.705625	547	6	5	7	5.3084375	991	2	8	0	1.665
104	10	1	4	2.91375	548	6	5	8	6.24625	992	2	8	1	2.08125
105	10	1	5	3.74625	549	6	5	9	6.24625	993	2	8	2	2.4975
106	10	1	6	4.370625	550	6	5	10	6.24625	994	2	8	3	2.91375
107	10	1	7	4.57875	551	6	4	0	2.08125	995	2	8	4	3.121875
108	10	1	8	5.41125	552	6	4	1	2.289375	996	2	8	5	4.1625
109	10	1	9	5.41125	553	6	4	2	2.3934375	997	2	8	6	4.57875
110	10	1	10	5.41125	554	6	4	3	2.91375	998	2	8	7	4.786875
111	10	0	0	1.665	555	6	4	4	3.0178125	999	2	8	8	5.8275
112	10	0	1	1.665	556	6	4	5	3.6421875	1000	2	8	9	5.8275
113	10	0	2	1.665	557	6	4	6	4.37125	1001	2	8	10	5.8275
114	10	0	3	2.4975	558	6	4	7	4.4753125	1002	2	7	0	1.665
115	10	0	4	2.4975	559	6	4	8	5.3084375	1003	2	7	1	2.08125
116	10	0	5	3.33	560	6	4	9	5.4125	1004	2	7	2	2.289375

117	10	0	6	4.1625	561	6	4	10	5.4125	1005	2	7	3	2.3934375
118	10	0	7	4.1625	562	6	3	0	2.08125	1006	2	7	4	2.6015625
119	10	0	8	4.995	563	6	3	1	2.1853125	1007	2	7	5	3.121875
120	10	0	9	4.995	564	6	3	2	2.289375	1008	2	7	6	3.6421875
121	10	0	10	4.995	565	6	3	3	2.86171875	1009	2	7	7	3.8503125
122	9	10	0	4.995	566	6	3	4	2.91375	1010	2	7	8	4.786875
123	9	10	1	5.41125	567	6	3	5	3.538125	1011	2	7	9	4.786875
124	9	10	2	5.8275	568	6	3	6	4.31921875	1012	2	7	10	4.786875
125	9	10	3	6.454375	569	6	3	7	4.37125	1013	2	6	0	1.665
126	9	10	4	6.6625	570	6	3	8	5.204375	1014	2	6	1	1.873125
127	9	10	5	7.4975	571	6	3	9	5.3084375	1015	2	6	2	2.08125
128	9	10	6	8.119375	572	6	3	10	5.3084375	1016	2	6	3	2.289375
129	9	10	7	8.3275	573	6	2	0	1.665	1017	2	6	4	2.3934375
130	9	10	8	9.1575	574	6	2	1	1.873125	1018	2	6	5	2.91375
131	9	10	9	9.57375	575	6	2	2	2.08125	1019	2	6	6	3.538125
132	9	10	10	9.57375	576	6	2	3	2.289375	1020	2	6	7	3.6421875
133	9	9	0	4.995	577	6	2	4	2.3934375	1021	2	6	8	4.57875
134	9	9	1	5.41125	578	6	2	5	2.91375	1022	2	6	9	4.57875
135	9	9	2	5.8275	579	6	2	6	3.538125	1023	2	6	10	4.57875
136	9	9	3	6.454375	580	6	2	7	3.6421875	1024	2	5	0	1.665
137	9	9	4	6.6625	581	6	2	8	4.57875	1025	2	5	1	2.08125
138	9	9	5	7.4975	582	6	2	9	4.57875	1026	2	5	2	2.4975
139	9	9	6	8.119375	583	6	2	10	4.57875	1027	2	5	3	2.08125
140	9	9	7	8.3275	584	6	1	0	1.665	1028	2	5	4	2.289375
141	9	9	8	9.1575	585	6	1	1	1.873125	1029	2	5	5	2.4975
142	9	9	9	9.57375	586	6	1	2	1.873125	1030	2	5	6	2.91375
143	9	9	10	9.57375	587	6	1	3	2.1853125	1031	2	5	7	3.121875
144	9	8	0	4.995	588	6	1	4	2.289375	1032	2	5	8	4.1625
145	9	8	1	5.41125	589	6	1	5	2.705625	1033	2	5	9	4.1625
146	9	8	2	5.8275	590	6	1	6	3.4340625	1034	2	5	10	4.1625
147	9	8	3	6.24625	591	6	1	7	3.538125	1035	2	4	0	0.8325
148	9	8	4	6.454375	592	6	1	8	4.370625	1036	2	4	1	1.24875
149	9	8	5	7.4975	593	6	1	9	4.370625	1037	2	4	2	1.456875
150	9	8	6	7.91125	594	6	1	10	4.370625	1038	2	4	3	1.5609375
151	9	8	7	8.119375	595	6	0	0	1.665	1039	2	4	4	1.7690625
152	9	8	8	9.1575	596	6	0	1	1.665	1040	2	4	5	2.289375
153	9	8	9	9.1575	597	6	0	2	1.665	1041	2	4	6	2.3934375
154	9	8	10	9.1575	598	6	0	3	2.08125	1042	2	4	7	2.6015625
155	9	7	0	4.1625	599	6	0	4	2.08125	1043	2	4	8	3.121875
156	9	7	1	4.57875	600	6	0	5	2.4975	1044	2	4	9	3.121875
157	9	7	2	4.786875	601	6	0	6	3.33	1045	2	4	10	3.121875
158	9	7	3	5.4125	602	6	0	7	3.33	1046	2	3	0	0.8325
159	9	7	4	5.620625	603	6	0	8	4.1625	1047	2	3	1	1.040625

160	9	7	5	6.454375	604	6	0	9	4.1625	1048	2	3	2	1.24875
161	9	7	6	7.07875	605	6	0	10	4.1625	1049	2	3	3	1.456875
162	9	7	7	7.286875	606	5	10	0	3.33	1050	2	3	4	1.5609375
163	9	7	8	8.119375	607	5	10	1	3.74625	1051	2	3	5	2.08125
164	9	7	9	8.3275	608	5	10	2	4.1625	1052	2	3	6	2.289375
165	9	7	10	8.3275	609	5	10	3	4.57875	1053	2	3	7	2.3934375
166	9	6	0	4.1625	610	5	10	4	4.786875	1054	2	3	8	2.91375
167	9	6	1	4.370625	611	5	10	5	5.8275	1055	2	3	9	2.91375
168	9	6	2	4.57875	612	5	10	6	6.24625	1056	2	3	10	2.91375
169	9	6	3	5.3084375	613	5	10	7	6.454375	1057	2	2	0	0
170	9	6	4	5.4125	614	5	10	8	7.4975	1058	2	2	1	0.41625
171	9	6	5	6.24625	615	5	10	9	7.4975	1059	2	2	2	0.8325
172	9	6	6	6.9746875	616	5	10	10	7.4975	1060	2	2	3	1.24875
173	9	6	7	7.07875	617	5	9	0	3.33	1061	2	2	4	1.456875
174	9	6	8	7.91125	618	5	9	1	3.74625	1062	2	2	5	2.4975
175	9	6	9	8.119375	619	5	9	2	4.1625	1063	2	2	6	2.08125
176	9	6	10	8.119375	620	5	9	3	4.57875	1064	2	2	7	2.289375
177	9	5	0	3.33	621	5	9	4	4.786875	1065	2	2	8	2.4975
178	9	5	1	3.74625	622	5	9	5	5.8275	1066	2	2	9	2.4975
179	9	5	2	4.1625	623	5	9	6	6.24625	1067	2	2	10	2.4975
180	9	5	3	4.57875	624	5	9	7	6.454375	1068	2	1	0	0
181	9	5	4	4.786875	625	5	9	8	7.4975	1069	2	1	1	0.41625
182	9	5	5	5.8275	626	5	9	9	7.4975	1070	2	1	2	0.41625
183	9	5	6	6.24625	627	5	9	10	7.4975	1071	2	1	3	1.040625
184	9	5	7	6.454375	628	5	8	0	3.33	1072	2	1	4	1.24875
185	9	5	8	7.4975	629	5	8	1	3.74625	1073	2	1	5	2.08125
186	9	5	9	7.4975	630	5	8	2	4.1625	1074	2	1	6	1.873125
187	9	5	10	7.4975	631	5	8	3	4.57875	1075	2	1	7	2.08125
188	9	4	0	2.4975	632	5	8	4	4.786875	1076	2	1	8	2.08125
189	9	4	1	2.91375	633	5	8	5	5.8275	1077	2	1	9	2.08125
190	9	4	2	3.121875	634	5	8	6	6.24625	1078	2	1	10	2.08125
191	9	4	3	3.74625	635	5	8	7	6.454375	1079	2	0	0	0
192	9	4	4	3.954375	636	5	8	8	7.4975	1080	2	0	1	0
193	9	4	5	4.786875	637	5	8	9	7.4975	1081	2	0	2	0
194	9	4	6	5.4125	638	5	8	10	7.4975	1082	2	0	3	0.8325
195	9	4	7	5.620625	639	5	7	0	2.4975	1083	2	0	4	0.8325
196	9	4	8	6.454375	640	5	7	1	2.91375	1084	2	0	5	1.665
197	9	4	9	6.6625	641	5	7	2	3.121875	1085	2	0	6	1.665
198	9	4	10	6.6625	642	5	7	3	3.6421875	1086	2	0	7	1.665
199	9	3	0	2.4975	643	5	7	4	3.8503125	1087	2	0	8	1.665
200	9	3	1	2.705625	644	5	7	5	4.786875	1088	2	0	9	1.665
201	9	3	2	2.91375	645	5	7	6	5.3084375	1089	2	0	10	1.665
202	9	3	3	3.6421875	646	5	7	7	5.5165625	1090	1	10	0	1.665

203	9	3	4	3.74625	647	5	7	8	6.454375	1091	1	10	1	2.08125
204	9	3	5	4.57875	648	5	7	9	6.454375	1092	1	10	2	2.08125
205	9	3	6	5.3084375	649	5	7	10	6.454375	1093	1	10	3	2.705625
206	9	3	7	5.4125	650	5	6	0	2.4975	1094	1	10	4	2.91375
207	9	3	8	6.24625	651	5	6	1	2.705625	1095	1	10	5	3.74625
208	9	3	9	6.454375	652	5	6	2	2.91375	1096	1	10	6	4.370625
209	9	3	10	6.454375	653	5	6	3	3.538125	1097	1	10	7	4.57875
210	9	2	0	1.665	654	5	6	4	3.6421875	1098	1	10	8	5.41125
211	9	2	1	2.08125	655	5	6	5	4.57875	1099	1	10	9	5.41125
212	9	2	2	2.4975	656	5	6	6	5.204375	1100	1	10	10	5.41125
213	9	2	3	2.91375	657	5	6	7	5.3084375	1101	1	9	0	1.665
214	9	2	4	3.121875	658	5	6	8	6.24625	1102	1	9	1	2.08125
215	9	2	5	4.1625	659	5	6	9	6.24625	1103	1	9	2	2.08125
216	9	2	6	4.57875	660	5	6	10	6.24625	1104	1	9	3	2.705625
217	9	2	7	4.786875	661	5	5	0	1.665	1105	1	9	4	2.91375
218	9	2	8	5.8275	662	5	5	1	2.08125	1106	1	9	5	3.74625
219	9	2	9	5.8275	663	5	5	2	2.4975	1107	1	9	6	4.370625
220	9	2	10	5.8275	664	5	5	3	2.91375	1108	1	9	7	4.57875
221	9	1	0	1.665	665	5	5	4	3.121875	1109	1	9	8	5.41125
222	9	1	1	2.08125	666	5	5	5	4.1625	1110	1	9	9	5.41125
223	9	1	2	2.08125	667	5	5	6	4.57875	1111	1	9	10	5.41125
224	9	1	3	2.705625	668	5	5	7	4.786875	1112	1	8	0	1.665
225	9	1	4	2.91375	669	5	5	8	5.8275	1113	1	8	1	2.08125
226	9	1	5	3.74625	670	5	5	9	5.8275	1114	1	8	2	2.08125
227	9	1	6	4.370625	671	5	5	10	5.8275	1115	1	8	3	2.705625
228	9	1	7	4.57875	672	5	4	0	1.665	1116	1	8	4	2.91375
229	9	1	8	5.41125	673	5	4	1	2.08125	1117	1	8	5	3.74625
230	9	1	9	5.41125	674	5	4	2	2.289375	1118	1	8	6	4.370625
231	9	1	10	5.41125	675	5	4	3	2.3934375	1119	1	8	7	4.57875
232	9	0	0	1.665	676	5	4	4	2.6015625	1120	1	8	8	5.41125
233	9	0	1	1.665	677	5	4	5	3.121875	1121	1	8	9	5.41125
234	9	0	2	1.665	678	5	4	6	3.6421875	1122	1	8	10	5.41125
235	9	0	3	2.4975	679	5	4	7	3.8503125	1123	1	7	0	1.665
236	9	0	4	2.4975	680	5	4	8	4.786875	1124	1	7	1	2.08125
237	9	0	5	3.33	681	5	4	9	4.786875	1125	1	7	2	2.08125
238	9	0	6	4.1625	682	5	4	10	4.786875	1126	1	7	3	2.289375
239	9	0	7	4.1625	683	5	3	0	1.665	1127	1	7	4	2.4975
240	9	0	8	4.995	684	5	3	1	1.873125	1128	1	7	5	2.91375
241	9	0	9	4.995	685	5	3	2	2.08125	1129	1	7	6	3.538125
242	9	0	10	4.995	686	5	3	3	2.289375	1130	1	7	7	3.74625
243	8	10	0	4.995	687	5	3	4	2.3934375	1131	1	7	8	4.57875
244	8	10	1	5.41125	688	5	3	5	2.91375	1132	1	7	9	4.57875
245	8	10	2	5.8275	689	5	3	6	3.538125	1133	1	7	10	4.57875

246	8	10	3	6.24625	690	5	3	7	3.6421875	1134	1	6	0	1.665
247	8	10	4	6.454375	691	5	3	8	4.57875	1135	1	6	1	1.873125
248	8	10	5	7.4975	692	5	3	9	4.57875	1136	1	6	2	1.873125
249	8	10	6	7.91125	693	5	3	10	4.57875	1137	1	6	3	2.1853125
250	8	10	7	8.119375	694	5	2	0	1.665	1138	1	6	4	2.289375
251	8	10	8	9.1575	695	5	2	1	2.08125	1139	1	6	5	2.705625
252	8	10	9	9.1575	696	5	2	2	2.4975	1140	1	6	6	3.4340625
253	8	10	10	9.1575	697	5	2	3	2.08125	1141	1	6	7	3.538125
254	8	9	0	4.995	698	5	2	4	2.289375	1142	1	6	8	4.370625
255	8	9	1	5.41125	699	5	2	5	2.4975	1143	1	6	9	4.370625
256	8	9	2	5.8275	700	5	2	6	2.91375	1144	1	6	10	4.370625
257	8	9	3	6.24625	701	5	2	7	3.121875	1145	1	5	0	1.665
258	8	9	4	6.454375	702	5	2	8	4.1625	1146	1	5	1	2.08125
259	8	9	5	7.4975	703	5	2	9	4.1625	1147	1	5	2	2.08125
260	8	9	6	7.91125	704	5	2	10	4.1625	1148	1	5	3	1.873125
261	8	9	7	8.119375	705	5	1	0	1.665	1149	1	5	4	2.08125
262	8	9	8	9.1575	706	5	1	1	2.08125	1150	1	5	5	2.08125
263	8	9	9	9.1575	707	5	1	2	2.08125	1151	1	5	6	2.705625
264	8	9	10	9.1575	708	5	1	3	1.873125	1152	1	5	7	2.91375
265	8	8	0	4.995	709	5	1	4	2.08125	1153	1	5	8	3.74625
266	8	8	1	5.41125	710	5	1	5	2.08125	1154	1	5	9	3.74625
267	8	8	2	5.8275	711	5	1	6	2.705625	1155	1	5	10	3.74625
268	8	8	3	6.24625	712	5	1	7	2.91375	1156	1	4	0	0.8325
269	8	8	4	6.454375	713	5	1	8	3.74625	1157	1	4	1	1.24875
270	8	8	5	7.4975	714	5	1	9	3.74625	1158	1	4	2	1.24875
271	8	8	6	7.91125	715	5	1	10	3.74625	1159	1	4	3	1.456875
272	8	8	7	8.119375	716	5	0	0	1.665	1160	1	4	4	1.665
273	8	8	8	9.1575	717	5	0	1	1.665	1161	1	4	5	2.08125
274	8	8	9	9.1575	718	5	0	2	1.665	1162	1	4	6	2.289375
275	8	8	10	9.1575	719	5	0	3	1.665	1163	1	4	7	2.4975
276	8	7	0	4.1625	720	5	0	4	1.665	1164	1	4	8	2.91375
277	8	7	1	4.57875	721	5	0	5	1.665	1165	1	4	9	2.91375
278	8	7	2	4.786875	722	5	0	6	2.4975	1166	1	4	10	2.91375
279	8	7	3	5.3084375	723	5	0	7	2.4975	1167	1	3	0	0.8325
280	8	7	4	5.5165625	724	5	0	8	3.33	1168	1	3	1	1.040625
281	8	7	5	6.454375	725	5	0	9	3.33	1169	1	3	2	1.040625
282	8	7	6	6.9746875	726	5	0	10	3.33	1170	1	3	3	1.3528125
283	8	7	7	7.1828125	727	4	10	0	2.4975	1171	1	3	4	1.456875
284	8	7	8	8.119375	728	4	10	1	2.91375	1172	1	3	5	1.873125
285	8	7	9	8.119375	729	4	10	2	3.121875	1173	1	3	6	2.1853125
286	8	7	10	8.119375	730	4	10	3	3.74625	1174	1	3	7	2.289375
287	8	6	0	4.1625	731	4	10	4	4.0584375	1175	1	3	8	2.705625
288	8	6	1	4.370625	732	4	10	5	4.786875	1176	1	3	9	2.705625

289	8	6	2	4.57875	733	4	10	6	5.4125	1177	1	3	10	2.705625
290	8	6	3	5.204375	734	4	10	7	5.7246875	1178	1	2	0	0
291	8	6	4	5.3084375	735	4	10	8	6.454375	1179	1	2	1	0.41625
292	8	6	5	6.24625	736	4	10	9	6.6625	1180	1	2	2	0.41625
293	8	6	6	6.870625	737	4	10	10	6.870625	1181	1	2	3	1.040625
294	8	6	7	6.9746875	738	4	9	0	2.4975	1182	1	2	4	1.24875
295	8	6	8	7.91125	739	4	9	1	2.91375	1183	1	2	5	2.08125
296	8	6	9	7.91125	740	4	9	2	3.121875	1184	1	2	6	1.873125
297	8	6	10	7.91125	741	4	9	3	3.74625	1185	1	2	7	2.08125
298	8	5	0	3.33	742	4	9	4	3.954375	1186	1	2	8	2.08125
299	8	5	1	3.74625	743	4	9	5	4.786875	1187	1	2	9	2.08125
300	8	5	2	4.1625	744	4	9	6	5.4125	1188	1	2	10	2.08125
301	8	5	3	4.57875	745	4	9	7	5.620625	1189	1	1	0	0
302	8	5	4	4.786875	746	4	9	8	6.454375	1190	1	1	1	0.41625
303	8	5	5	5.8275	747	4	9	9	6.6625	1191	1	1	2	0.41625
304	8	5	6	6.24625	748	4	9	10	6.6625	1192	1	1	3	1.040625
305	8	5	7	6.454375	749	4	8	0	2.4975	1193	1	1	4	1.24875
306	8	5	8	7.4975	750	4	8	1	2.91375	1194	1	1	5	2.08125
307	8	5	9	7.4975	751	4	8	2	3.121875	1195	1	1	6	1.873125
308	8	5	10	7.4975	752	4	8	3	3.6421875	1196	1	1	7	2.08125
309	8	4	0	2.4975	753	4	8	4	3.8503125	1197	1	1	8	2.08125
310	8	4	1	2.91375	754	4	8	5	4.786875	1198	1	1	9	2.08125
311	8	4	2	3.121875	755	4	8	6	5.3084375	1199	1	1	10	2.08125
312	8	4	3	3.6421875	756	4	8	7	5.5165625	1200	1	0	0	0
313	8	4	4	3.8503125	757	4	8	8	6.454375	1201	1	0	1	0
314	8	4	5	4.786875	758	4	8	9	6.454375	1202	1	0	2	0
315	8	4	6	5.3084375	759	4	8	10	6.454375	1203	1	0	3	0.8325
316	8	4	7	5.5165625	760	4	7	0	2.08125	1204	1	0	4	0.8325
317	8	4	8	6.454375	761	4	7	1	2.4975	1205	1	0	5	1.665
318	8	4	9	6.454375	762	4	7	2	2.6015625	1206	1	0	6	1.665
319	8	4	10	6.454375	763	4	7	3	3.0178125	1207	1	0	7	1.665
320	8	3	0	2.4975	764	4	7	4	3.27796875	1208	1	0	8	1.665
321	8	3	1	2.705625	765	4	7	5	3.8503125	1209	1	0	9	1.665
322	8	3	2	2.91375	766	4	7	6	4.4753125	1210	1	0	10	1.665
323	8	3	3	3.538125	767	4	7	7	4.73546875	1211	0	10	0	1.665
324	8	3	4	3.6421875	768	4	7	8	5.5165625	1212	0	10	1	1.665
325	8	3	5	4.57875	769	4	7	9	5.620625	1213	0	10	2	1.665
326	8	3	6	5.204375	770	4	7	10	5.7246875	1214	0	10	3	2.4975
327	8	3	7	5.3084375	771	4	6	0	2.08125	1215	0	10	4	2.4975
328	8	3	8	6.24625	772	4	6	1	2.289375	1216	0	10	5	3.33
329	8	3	9	6.24625	773	4	6	2	2.3934375	1217	0	10	6	4.1625
330	8	3	10	6.24625	774	4	6	3	2.91375	1218	0	10	7	4.1625
331	8	2	0	1.665	775	4	6	4	3.0178125	1219	0	10	8	4.995

332	8	2	1	2.08125	776	4	6	5	3.6421875	1220	0	10	9	4.995
333	8	2	2	2.4975	777	4	6	6	4.37125	1221	0	10	10	4.995
334	8	2	3	2.91375	778	4	6	7	4.4753125	1222	0	9	0	1.665
335	8	2	4	3.121875	779	4	6	8	5.3084375	1223	0	9	1	1.665
336	8	2	5	4.1625	780	4	6	9	5.4125	1224	0	9	2	1.665
337	8	2	6	4.57875	781	4	6	10	5.4125	1225	0	9	3	2.4975
338	8	2	7	4.786875	782	4	5	0	1.665	1226	0	9	4	2.4975
339	8	2	8	5.8275	783	4	5	1	2.08125	1227	0	9	5	3.33
340	8	2	9	5.8275	784	4	5	2	2.289375	1228	0	9	6	4.1625
341	8	2	10	5.8275	785	4	5	3	2.3934375	1229	0	9	7	4.1625
342	8	1	0	1.665	786	4	5	4	2.6015625	1230	0	9	8	4.995
343	8	1	1	2.08125	787	4	5	5	3.121875	1231	0	9	9	4.995
344	8	1	2	2.08125	788	4	5	6	3.6421875	1232	0	9	10	4.995
345	8	1	3	2.705625	789	4	5	7	3.8503125	1233	0	8	0	1.665
346	8	1	4	2.91375	790	4	5	8	4.786875	1234	0	8	1	1.665
347	8	1	5	3.74625	791	4	5	9	4.786875	1235	0	8	2	1.665
348	8	1	6	4.370625	792	4	5	10	4.786875	1236	0	8	3	2.4975
349	8	1	7	4.57875	793	4	4	0	1.24875	1237	0	8	4	2.4975
350	8	1	8	5.41125	794	4	4	1	1.665	1238	0	8	5	3.33
351	8	1	9	5.41125	795	4	4	2	1.7690625	1239	0	8	6	4.1625
352	8	1	10	5.41125	796	4	4	3	1.9771875	1240	0	8	7	4.1625
353	8	0	0	1.665	797	4	4	4	2.23734375	1241	0	8	8	4.995
354	8	0	1	1.665	798	4	4	5	2.6015625	1242	0	8	9	4.995
355	8	0	2	1.665	799	4	4	6	3.0178125	1243	0	8	10	4.995
356	8	0	3	2.4975	800	4	4	7	3.27796875	1244	0	7	0	1.665
357	8	0	4	2.4975	801	4	4	8	3.8503125	1245	0	7	1	1.665
358	8	0	5	3.33	802	4	4	9	3.954375	1246	0	7	2	1.665
359	8	0	6	4.1625	803	4	4	10	4.0584375	1247	0	7	3	2.08125
360	8	0	7	4.1625	804	4	3	0	1.24875	1248	0	7	4	2.08125
361	8	0	8	4.995	805	4	3	1	1.456875	1249	0	7	5	2.4975
362	8	0	9	4.995	806	4	3	2	1.5609375	1250	0	7	6	3.33
363	8	0	10	4.995	807	4	3	3	1.873125	1251	0	7	7	3.33
364	7	10	0	4.1625	808	4	3	4	1.9771875	1252	0	7	8	4.1625
365	7	10	1	4.57875	809	4	3	5	2.3934375	1253	0	7	9	4.1625
366	7	10	2	4.786875	810	4	3	6	2.91375	1254	0	7	10	4.1625
367	7	10	3	5.4125	811	4	3	7	3.0178125	1255	0	6	0	1.665
368	7	10	4	5.7246875	812	4	3	8	3.6421875	1256	0	6	1	1.665
369	7	10	5	6.454375	813	4	3	9	3.74625	1257	0	6	2	1.665
370	7	10	6	7.07875	814	4	3	10	3.74625	1258	0	6	3	2.08125
371	7	10	7	7.3909375	815	4	2	0	0.8325	1259	0	6	4	2.08125
372	7	10	8	8.119375	816	4	2	1	1.24875	1260	0	6	5	2.4975
373	7	10	9	8.3275	817	4	2	2	1.456875	1261	0	6	6	3.33
374	7	10	10	8.535625	818	4	2	3	1.5609375	1262	0	6	7	3.33

375	7	9	0	4.1625	819	4	2	4	1.7690625	1263	0	6	8	4.1625
376	7	9	1	4.57875	820	4	2	5	2.289375	1264	0	6	9	4.1625
377	7	9	2	4.786875	821	4	2	6	2.3934375	1265	0	6	10	4.1625
378	7	9	3	5.4125	822	4	2	7	2.6015625	1266	0	5	0	1.665
379	7	9	4	5.620625	823	4	2	8	3.121875	1267	0	5	1	1.665
380	7	9	5	6.454375	824	4	2	9	3.121875	1268	0	5	2	1.665
381	7	9	6	7.07875	825	4	2	10	3.121875	1269	0	5	3	1.665
382	7	9	7	7.286875	826	4	1	0	0.8325	1270	0	5	4	1.665
383	7	9	8	8.119375	827	4	1	1	1.24875	1271	0	5	5	1.665
384	7	9	9	8.3275	828	4	1	2	1.24875	1272	0	5	6	2.4975
385	7	9	10	8.3275	829	4	1	3	1.456875	1273	0	5	7	2.4975
386	7	8	0	4.1625	830	4	1	4	1.665	1274	0	5	8	3.33
387	7	8	1	4.57875	831	4	1	5	2.08125	1275	0	5	9	3.33
388	7	8	2	4.786875	832	4	1	6	2.289375	1276	0	5	10	3.33
389	7	8	3	5.3084375	833	4	1	7	2.4975	1277	0	4	0	0.8325
390	7	8	4	5.5165625	834	4	1	8	2.91375	1278	0	4	1	0.8325
391	7	8	5	6.454375	835	4	1	9	2.91375	1279	0	4	2	0.8325
392	7	8	6	6.9746875	836	4	1	10	2.91375	1280	0	4	3	1.24875
393	7	8	7	7.1828125	837	4	0	0	0.8325	1281	0	4	4	1.24875
394	7	8	8	8.119375	838	4	0	1	0.8325	1282	0	4	5	1.665
395	7	8	9	8.119375	839	4	0	2	0.8325	1283	0	4	6	2.08125
396	7	8	10	8.119375	840	4	0	3	1.24875	1284	0	4	7	2.08125
397	7	7	0	3.33	841	4	0	4	1.24875	1285	0	4	8	2.4975
398	7	7	1	3.74625	842	4	0	5	1.665	1286	0	4	9	2.4975
399	7	7	2	3.8503125	843	4	0	6	2.08125	1287	0	4	10	2.4975
400	7	7	3	4.4753125	844	4	0	7	2.08125	1288	0	3	0	0.8325
401	7	7	4	4.73546875	845	4	0	8	2.4975	1289	0	3	1	0.8325
402	7	7	5	5.5165625	846	4	0	9	2.4975	1290	0	3	2	0.8325
403	7	7	6	6.1415625	847	4	0	10	2.4975	1291	0	3	3	1.24875
404	7	7	7	6.40171875	848	3	10	0	2.4975	1292	0	3	4	1.24875
405	7	7	8	7.1828125	849	3	10	1	2.705625	1293	0	3	5	1.665
406	7	7	9	7.286875	850	3	10	2	2.91375	1294	0	3	6	2.08125
407	7	7	10	7.3909375	851	3	10	3	3.6421875	1295	0	3	7	2.08125
408	7	6	0	3.33	852	3	10	4	3.74625	1296	0	3	8	2.4975
409	7	6	1	3.538125	853	3	10	5	4.57875	1297	0	3	9	2.4975
410	7	6	2	3.6421875	854	3	10	6	5.3084375	1298	0	3	10	2.4975
411	7	6	3	4.37125	855	3	10	7	5.4125	1299	0	2	0	0
412	7	6	4	4.4753125	856	3	10	8	6.24625	1300	0	2	1	0
413	7	6	5	5.3084375	857	3	10	9	6.454375	1301	0	2	2	0
414	7	6	6	6.0375	858	3	10	10	6.454375	1302	0	2	3	0.8325
415	7	6	7	6.1415625	859	3	9	0	2.4975	1303	0	2	4	0.8325
416	7	6	8	6.9746875	860	3	9	1	2.705625	1304	0	2	5	1.665
417	7	6	9	7.07875	861	3	9	2	2.91375	1305	0	2	6	1.665

418	7	6	10	7.07875	862	3	9	3	3.6421875	1306	0	2	7	1.665
419	7	5	0	2.4975	863	3	9	4	3.74625	1307	0	2	8	1.665
420	7	5	1	2.91375	864	3	9	5	4.57875	1308	0	2	9	1.665
421	7	5	2	3.121875	865	3	9	6	5.3084375	1309	0	2	10	1.665
422	7	5	3	3.6421875	866	3	9	7	5.4125	1310	0	1	0	0
423	7	5	4	3.8503125	867	3	9	8	6.24625	1311	0	1	1	0
424	7	5	5	4.786875	868	3	9	9	6.454375	1312	0	1	2	0
425	7	5	6	5.3084375	869	3	9	10	6.454375	1313	0	1	3	0.8325
426	7	5	7	5.5165625	870	3	8	0	2.4975	1314	0	1	4	0.8325
427	7	5	8	6.454375	871	3	8	1	2.705625	1315	0	1	5	1.665
428	7	5	9	6.454375	872	3	8	2	2.91375	1316	0	1	6	1.665
429	7	5	10	6.454375	873	3	8	3	3.538125	1317	0	1	7	1.665
430	7	4	0	2.08125	874	3	8	4	3.6421875	1318	0	1	8	1.665
431	7	4	1	2.4975	875	3	8	5	4.57875	1319	0	1	9	1.665
432	7	4	2	2.6015625	876	3	8	6	5.204375	1320	0	1	10	1.665
433	7	4	3	3.0178125	877	3	8	7	5.3084375	1321	0	0	0	0
434	7	4	4	3.27796875	878	3	8	8	6.24625	1322	0	0	1	0
435	7	4	5	3.8503125	879	3	8	9	6.24625	1323	0	0	2	0
436	7	4	6	4.4753125	880	3	8	10	6.24625	1324	0	0	3	0.8325
437	7	4	7	4.73546875	881	3	7	0	2.08125	1325	0	0	4	0.8325
438	7	4	8	5.5165625	882	3	7	1	2.289375	1326	0	0	5	1.665
439	7	4	9	5.620625	883	3	7	2	2.3934375	1327	0	0	6	1.665
440	7	4	10	5.7246875	884	3	7	3	2.91375	1328	0	0	7	1.665
441	7	3	0	2.08125	885	3	7	4	3.0178125	1329	0	0	8	1.665
442	7	3	1	2.289375	886	3	7	5	3.6421875	1330	0	0	9	1.665
443	7	3	2	2.3934375	887	3	7	6	4.37125	1331	0	0	10	1.665
444	7	3	3	2.91375	888	3	7	7	4.4753125					

11. Appendix-C

11.1 Prototype Implementation of SLAAgent

The prototype implementation of SLAAgent is given in this appendix. The implementation is based on the list of services available in the UDDI, Requirement Processor, SLA Manager and QoS Manager. The prototype has been implemented using the Microsoft Visual Basic 6 on front end and Microsoft Access 2007 Database on the back end. The functional details of the sub-components of SLAAgent along with pictorial representation is given in the following sections.

11.1.1 SLAAgent Main Interface

The SLAAgent main user interface is shown in Figure 11-1 which covers the high-level framework components shown in Chapter 5 (Figure 5-2) using the techniques of Fuzzy Logic for QoS calculation.

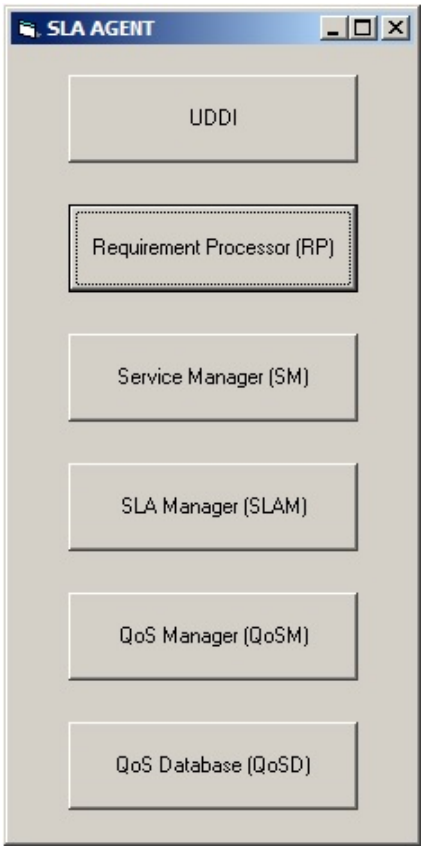

Figure 11-1: SLAAgent Main User Interface

11.1.2 Universal Description Discovery and Integration

The Universal Description Discovery and Integration (UDDI) also known as repository of the service providers with service descriptions is shown in Figure 11-2. There are 110 sample services with service provider information for Web Hosting Server providers and 110 services for File Storage Server providers are stored in the back end database. The database fields are based on technical description of the services, QoS information calculated from two selected QoS metrics and the price of the service.

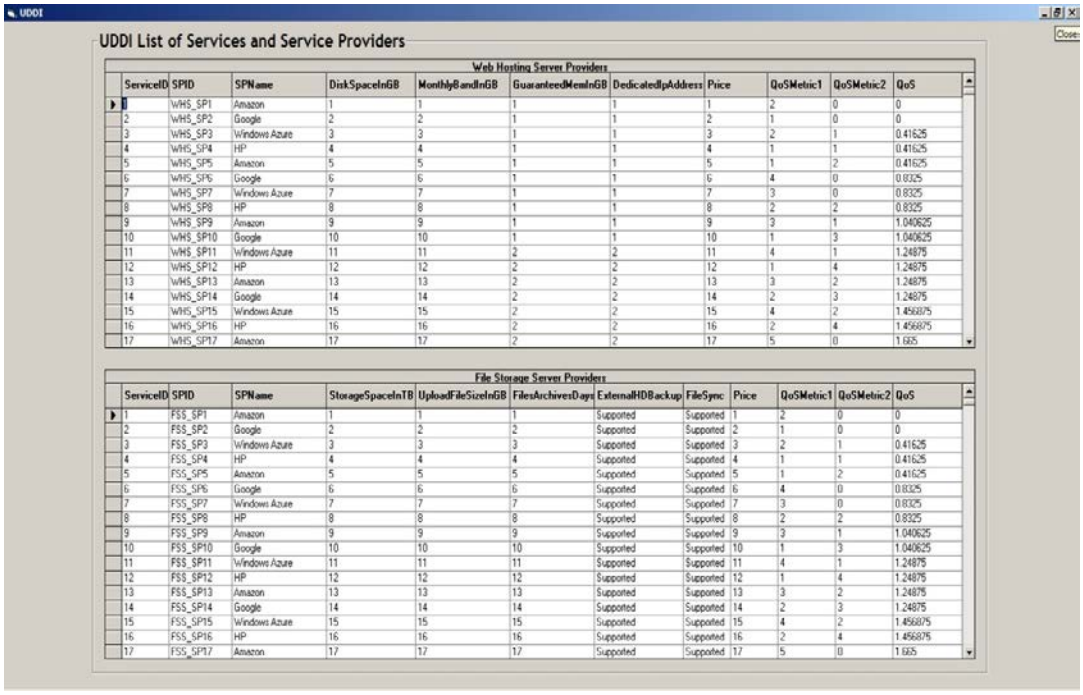

Figure 11-2: UDDI (Repository of Services and Service Providers)

11.1.3 Requirement Processor (RP)

The Requirement Processor (RP) interface is used to get the user requirements from the consumer which is shown in Figure 11-3. The consumer can search Web Hosting Services and File Storage Services by providing the minimum and maximum values. The search parameters used for Web Hosting Server are: Disk Space, Monthly Bandwidth and Price. The search parameters used for File Storage Server are: Storage Space, File Upload Size and Price. A list of auto filled sample inputs is also provided to quickly fill the required parameters for searching the services. Once the service providers of Web Hosting and File Storage are found, the Composite plans are

generated. The composite plans generated can be filtered further according to the Consumer preferences. The composite plans can be sorted in descending or ascending order with respect to individual QoS scores of service providers and total cost of the composite plan. The consumer can also alter the search parameters to regenerate the composite plans in order to narrow or widen the search requirements.

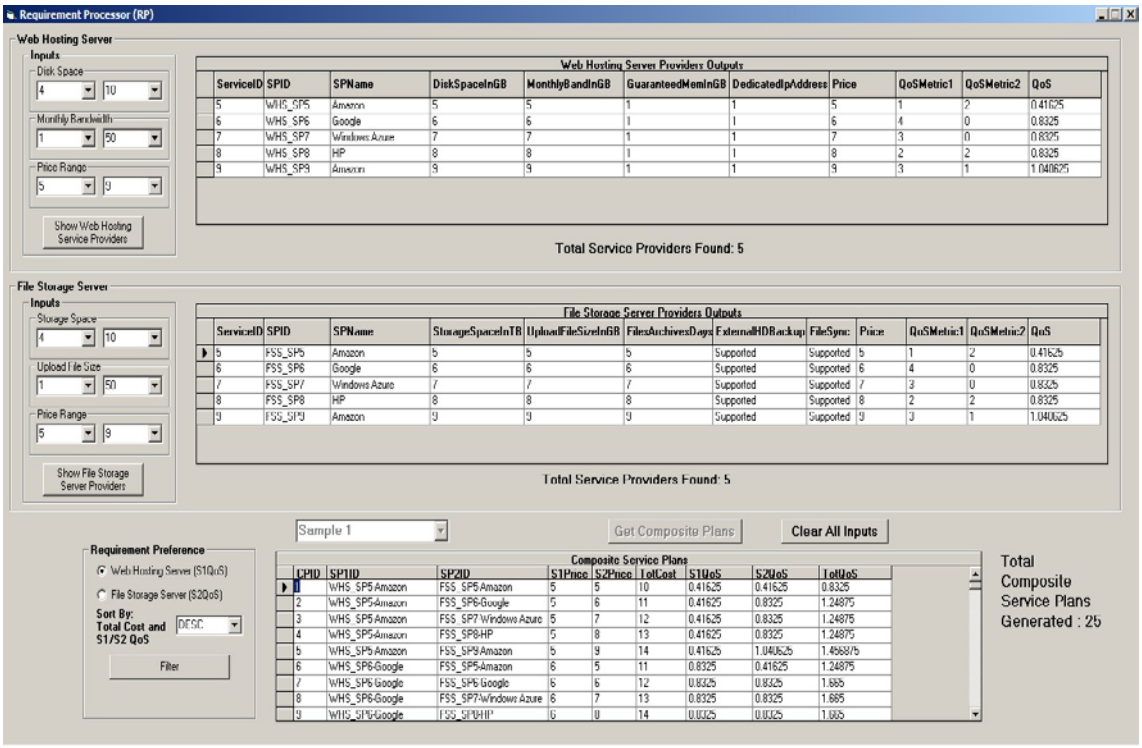

Figure 11-3: Requirement Processor (RP)

11.1.4 SLA Manager (SLAM)

The SLA Manager (SLAM) is used to generate the SLAs related with particular services from different service providers shown in Figure 11-4. The structure of SLAs is based on the SLA elements taken from Chapter 2 (Table 2-6).

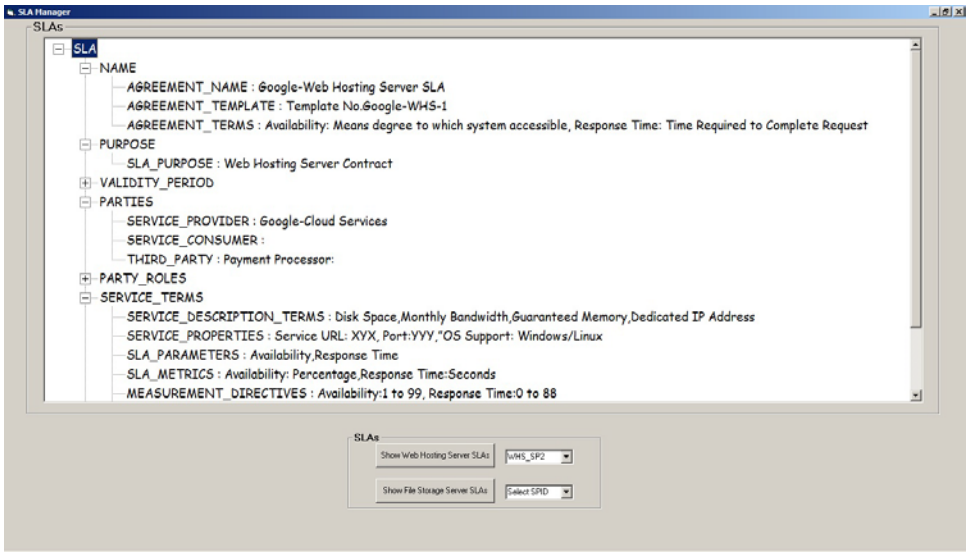

Figure 11-4: SLAs Generated from SLA Manager

11.1.5 QoS Manager (QoSM)

The QoS Manager is used to calculate the Quality of Service for service providers based on QoS metrics using Fuzzy Inference Technique shown in Figure 11-5. The Input Membership Function and Output Membership Function Graphs are provided for understanding the relationship between the Inputs and Output. A mockup calculation table based on two inputs and one output is generated. A fuzzy Inference calculator for calculating QoS between any two input values also provided.

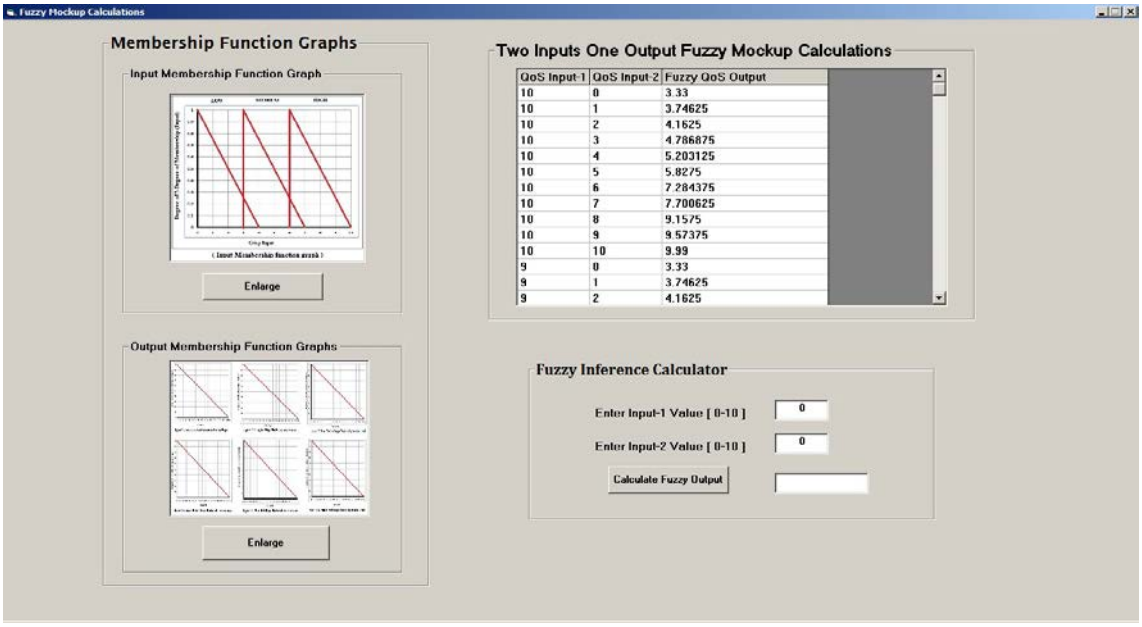

Figure 11-5: QoS Manager