Approaches for Discovery, Selection and Composition for Heterogeneous Web services using Semantic Web

A Thesis Submitted To

Nirma University

In Partial Fulfillment Of The Requirements For

The Degree Of

Doctor Of Philosophy

In

Technology & Engineering

By

Mr Kiritkumar Jayantilal Modi (10EXTPHDE32)

Institute of Technology
Nirma University
Ahmedabad-382481
Gujarat, India
February 2016

Nirma University
Institute of Technology
Certificate

This is to certify that the thesis entitled Approaches for Discovery, Selection and Composition for Heterogeneous Web services using Semantic Web has been prepared by Mr Kiritkumar Jayantilal Modi under my supervision and guidance. The thesis is his original work completed after careful research and investigation. The work of the thesis is of the standard expected of a candidate for Ph.D. Programme in Computer Science and Engineering and I recommend that it be sent for evaluation.

Date: 25/2/16

[illegible]
S. Garg
Signature of the Guide

Forwarded Through:

[illegible]
S. Garg
Name and Signature of the

Head of the Department

[illegible] 25-2-16
Dr. P. N. Tekwani

Name and Signature of the

Dean Faculty of Technology

& Engineering

[illegible] 26/2/2016.

Name and Signature of the

Dean Faculty of Doctoral Studies

and Research

Executive Registrar

Abstract

A significant amount of research has been progressing to discover, select and compose heterogeneous Web services, including, SOAP, RESTful and Cloud services to fulfill the user request due to the availability and wide acceptance of the Web services. Use of Semantic Web plays a key role to make the service discovery, selection and composition, dynamic and automated by reducing the intervention of the end user. Traditional SOAP services increases XML processing overhead at the description and communication level. This issue could be resolved by RESTful services, which are based on the REST (REpresentational State Transfer) architecture, which makes it more suitable compared to the SOAP services. Meanwhile, cloud services has gained the importance because the cloud computing provides an efficient, on-demand and scalable platform to integrate the resources over the Web from the multiple sources.

Very few efforts have been made to develop solutions to perform the service discovery, selection and composition processes in an integrated manner using the Semantic Web. An integrated approach for service discovery, selection and composition using the domain ontology and QoS support for SOAP services has been proposed by me. I have developed a prototype for the Healthcare Information System (HIS) using the proposed approach of SOAP service. For the RESTful services, to perform the service discovery, selection and composition, Linked Open Data-based approach has been proposed by me by considering the QoS parameters (i.e., throughput and response time) and semantic description(i.e. RDF data). Prototypes for the Population Information System(PIS) and Healthcare Recommendation System(HRS) have been developed to demonstrate the effectiveness of the proposed work. I have proposed framework and approaches to perform the cloud service discovery, selection and composition using cloud ontology. Based on the proposed framework, the prototype for Healthcare Decision Making System (HDMS) is developed by me. The experimental results show that the proposed approaches have demonstrated effective performance improvement in comparison with existing QoS-based and Semantic Web-based approaches. The proposed solution, fulfills the need of an end user to search, select and compose heterogeneous Web services with increased level of user satisfaction and higher degree of automation.

Nirma University
Institute of Technology
Declaration

I, Mr Kiritkumar Jayantilal Modi, registered as Research Scholar, bearing Registration No.10EXTPHDE32 for Doctoral Programme under the Faculty of Technology & Engineering of Nirma University do hereby declare that I have completed the course work, pre-synopsis seminar and my research work as prescribed under R.Ph.D. 3.5.

I do hereby declare that the thesis submitted is original and is the outcome of the independent investigations / research carried out by me and contains no plagiarism. The research is leading to the discovery of new techniques already known. This work has not been submitted by any other University or Body in quest of a degree, diploma or any other kind of academic award.

I do hereby further declare that the text, diagrams or any other material taken from other sources (including but not limited to books, journals and web) have been acknowledged, referred and cited to the best of my knowledge and understanding.

Date: 25/2/16

[illegible]

Signature of Student

Acknowledgements

I would like to thank Dr. Sanjay Garg, Guide and the Head of Computer Science and Engineering Department, Institute of Technology, Nirma University, Ahmedabad for motivating me and providing continous support throughout my Doctoral studies.

My sincere thanks to the reviewers Dr. Sanjay Chaudhary and Dr. K Kotecha, who had given their valuable feedback during my Research Progress Committee meetings.

I am also thankful to all faculty members, staff members of the department for providing all types of resources and support in time.

There are no words to thank the Almighty for gifting me with wonderful children Dhairya and Dhruv, who bore the wrath of my journey towards this research. My wife Priti needs special accolade for her patience and constant support provided throughout this work. Finally, my most sincere thanks to the most important and special people of my life - My Parents; who have been ever motivating, endearing and highly cooperative in all my endeavors.

I thank one and all, who have kept encouraging and motivating me. Without everyone's support, this thesis is beyond imagination.

Mr Kiritkumar Modi

10EXTPHDE32

Contents

List of Tables

List of Figures

Abbreviations

AI	Artificial Intelligence
BPEL	Business Process Execution Language
DAG	Directed Acyclic Graph
DAML-S	DARPA Agent Markup Language for Services
HATEOAS	Hypermedia as the Engine of Application State
HTML	HyperText Markup Language
HTTP	Hyper Text Transfer Protocol
OASIS	Organization for the Advancement of Structured Information Standards
OWL	Ontology Web Language
OWL-S	Web Ontology Language for Services
QoS	Quality of Service
RDF	Resource Description Framework
RDFS	RDF Schema
REST	Representation State Transfer
ROA	Resource Oriented Architecture
SAWSDL	Semantic Annotations for WSDL and XML Schema
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SOC	Service-oriented Computing
SPARQL	SPARQL Protocol and RDF Query Language
SUPER	Semantics Utilized for Process management within and between EnteRprises
SWS	Semantic Web service
UDDI	Universal Description Discovery and Integration

USDL	Universal Service-semantics Description Language
WADL	Web Application Description Language
WS	Web Service
WS-BPEL	Web Services Business Process Execution Language
WS-CDL	Web Services Choreography Description Language
WSCI	Web Service Choreography Interface
WSDL	Web Service Description Language
WSFL	Web Service Flow Language
WSMO	Web Service Modeling Ontology
WSMX	Web Service Modeling eXecution Environment
W3C	World Wide Web Consortium
XML	Extensible Markup Language

Chapter 1

Introduction

The Web has become a common platform to help a user to search and perform different tasks, such as purchasing electronic items and taking an appointment from medical practitioner. With increased growth of services, it has become difficult for a user to search and select the required services. If a single service is not able to fulfill the complete requirement, it become necessary to combine multiple services to get the desired result. Organizations like, Google, Microsoft, and Amazon have established to deliver their resources as Web services over the Internet. Furthermore, the rapid growth of Cloud computing and Internet of Things (IoT) has increased the demand and re-usability of the Web services (Armbrust et al.), (Atzori, Iera, and Morabito).

Web services may be atomic or composite (Benatallah, Dumas, and Maamar). An atomic service or an elementary service does not depend upon another Web service to fulfill the user requirement, while a composite service integrates other atomic and composite services to offer a value added service to the user. A composite service is represented as a collection of the component services. For example, a composite Web service Travel Planner may aggregate multiple Web services for flight booking, travel insurance, accommodation booking, car rental, etc., which may be executed sequentially or concurrently (Benatallah, Dumas, and Maamar). In practice, organizations use different styles to describe and develop the Web services. Nowadays, there are two types of Web services that are commonly in use: (i) SOAP-based Web services and (ii) RESTful Web services (Pautasso, Zimmermann, and Leymann),(Adamczyk et al.). The first one is based on Web Services Description Language (WSDL) (Christensen et al.) and Simple Object Access Protocol (SOAP) (Gudgin et al.), while the second one follows the REST principles (Fielding). Moreover, end users are able to access these two types of Web services on demand from the cloud service providers, over the Web. This is known as Cloud services. These variants of Web services defines the term Heterogeneous Web services. Heterogeneous services refer to services, that have different functional aspects, such as the interface, the implementation, the data, etc (Neupane et al.). (Lee, Lee, and Wang) have presented SOAP-based Web services, RESTful Web services and Non-Web services as Heterogeneous Web services in the research work.

1.1 Heterogeneous Web services

A brief introduction of Heterogeneous Web services and concepts associated to service discovery, selection and composition are presented as follows.

I. **SOAP-based Web services**

Web services has been considered as the preferred way to implement the Service Oriented Architecture (SOA) ([Papazoglou et al.), (Papazoglou et al.), (Sheng et al.), (Papazoglou and Van Den Heuvel) and its associated set of objectives. A Web service is a software component designed to support inter-operable application-to-application interactions over the network (Booth et al.). It has a WSDL interface, which can be published and possibly found from Universal Description Discovery and Integration (UDDI) (Bellwood et al.) and accessed through SOAP Protocol. It is important to note that WSDL document can be accessed without the need of UDDI. The Web services architecture (Booth et al.) proposed by W3C has three entities:(i) Web service requester,(ii)Web service Provider, and (iii)Web service Registry, where requester uses the Web services offered by the providers and registry allows these entities to publish and find the Web services. WSDL, SOAP, and UDDI are the three core standards of Web services technologies, which are based on XML language. WSDL is an XML-based interface for describing the Web services. SOAP is an XML-based protocol for communication between the Web services. While, UDDI is the registry standard, where Web services can be published and found by other entities. More information on Web services has been provided by (Gustavo et al.) and (Newcomer).

II. RESTful Web services

The Resources provided on the Web are increasingly used by applications. REST (REpresentational State Transfer) has gained significant importance, which follows Resource Oriented Architecture(ROA). REST is a set of architectural constraints defined by (Fielding), and these are used to implement RESTful Web services (Pautasso, Zimmermann, and Leymann). REST defines the interaction between a client and a server as the manipulation of states of URI-identified resources with a constrained set of operations, i.e., the HTTP methods. Moreover, hypermedia controls (i.e., links to other resources) allow clients to navigate from one resource to another during their interaction. RESTful Web services is a lightweight compared to the SOAP-based standards. REST is an architectural style, where distributed systems are built on a shared model and have agreement between nouns (resource names as URIs), verbs (HTTP methods used) and content types (usually XML or SOAP).

III. Cloud Services

Cloud services have changed the way computational resources is delivered to customers, by offering computing and storage capacity from remote data centers on demand, i.e., Web services offered as a utility (Drago). A cloud service (Shawish and Salama) is defined as the service, which is made available to the users, on-demand, via the Internet from the cloud provider. All the fundamental services that applied as cloud services are based on the Web services standards. However, creating, offering, selecting services efficiently over the cloud is an open challenge. Cloud computing has been implemented as a major service-oriented paradigm, delivering numerous information technology resources in Web-based services, such as Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS). With the increased use of Cloud services, service discovery, selection and composition to fulfill the user requirements has become a critical problem related to Cloud services research.

1.1.1 Discovery, Selection and Composition of Heterogeneous Web services

The Web services composition life cycle is proposed by (Sheng et al.), (Moghaddam and Davis), (Shehu, Epiphaniou, and Safdar), (Upadhyaya) as shown in Figure 1.1. My contribution focuses on three main phases of the life cycle: (i)Service Discovery, (ii) Service Selection and (iii) Service Composition for the Heterogeneous Web services.

A service provider publishes the service on a Web service registry, which contains different kinds of service descriptions, i.e., Web Service Description Language (WSDL) or Web Application Description Language (WADL) (Hadley). A service customer queries the registry to retrieve services. Service descriptions specify capabilities of services and usually include inputs, outputs, functional and non-functional description. Figure 1.1 shows the various stages involved in the life cycle of service composition, which are described as follows.

- **Service Discovery**
  Service discovery is a process of searching for services based on user requirements. Typically, a service discovery requires performing the matching operation between the capabilities of services published by the services providers in the service repository and a requester queries. An intermediate entity could be the reasoner in this matchmaking process, which will return the matched result.

- **Service Selection**
  Service selection is the process of selecting (as well as filtering) a suitable service from a pool of functionally equivalent services. Most of the research in service selection is based on Quality of Services (QoS) incorporated in a selection algorithm to achieve an optimized solution. Generally, users express their preferences using QoS parameters. QoS-based techniques help a user to select a service from the results

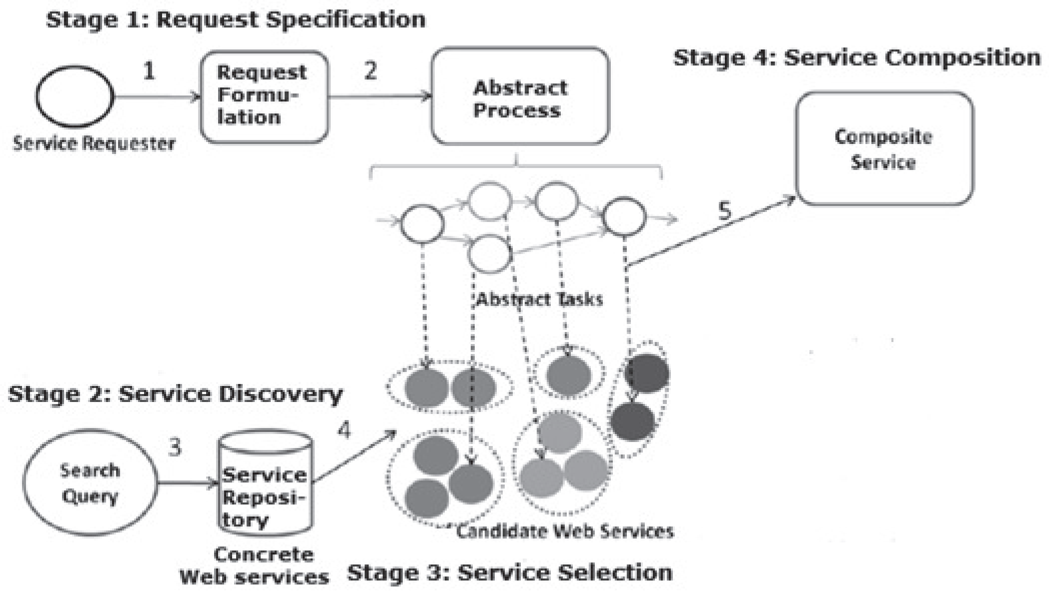


Figure 1.1: The Lifecycle of Web Service Composition

of functionally equivalent services. Reputation and trust-based mechanisms (Liu, Ngu, and Zeng), (Maximilien and Singh) are also used to filter best services from the service pool.

- **Service Composition**

  Service composition is defined as an aggregation of elementary and composite services. The service composition allows a user to create value added services on top of service description, discovery, and selection activities. Composite services offer reusable capabilities to developers. Service composition provides a seamless access to a variety of complex services to a user. Control flow information, which defines the order of execution of services, is maintained in the form of composition plan (also known as an abstract process). In this process, data flow will be performed, which helps to identify reusable data inputs among services.

1.1.2 Exploiting Heterogeneous Web services in Healthcare Domain

An ideal solution is required to make healthcare information inter-operable and easily accessible without any interference of the human-being. By considering this objective, I have proposed a solution for Web services based Healthcare Information System(HIS) using Semantic Web for the Dentistry domain. My objective of the proposed approach is to provide the platform to the users to search, select and compose healthcare services automatically with easy access and increased level of satisfaction. For the emergency medical situation, effective data management methods and tools are required to move from a web of documents (only understandable by human users) to a web of data, in which information is expressed in a format that can be read and used by machines as well. This would enables to find, share, and integrate information more easily. A prototype has been developed by me using proposed approach for RESTful Web services for Healthcare Recommendation System to help the end user to search, select and compose services using Public data. The exchange and integration of online medical information managed by several healthcare organizations and test centres becomes costly and complex. This motivates to use the cloud computing services and applications in the healthcare domain to alleviate the above problems. A prototype has been developed by me using Cloud services-based approaches for the Healthcare Decision Making System(HDMS) to improve the collaboration among various healthcare organizations and to deal with the challenges, like scalability, availability, throughput, response time, cost effectiveness that are related with Cloud services.

1.2 Motivation

Nowadays, Web services are considered the most demanding mechanism of distributed computing. Moreover, service Discovery and selection are the fundamental processes to search and select composable services based on semantic and non-functional description apart from functional and behavioral aspects to allow rapid creation of new Web services from existing services.

Service discovery, selection and composition are in general, complex tasks, that require considerable effort especially, when vast amounts of services are available. Web service discovery, selection and composition should be performed in an integrated manner due to the strong dependency among these tasks (Rodriguez Mier et al.). However, researchers have considered the service discovery, selection and composition tasks at individual level. This makes the solution inefficient or partially efficient.

In (Rodriguez Mier et al.), it is clearly identified that, the discovery and composition of Web services should be based on semantic description and should also be QOS-aware. In other words, it states that the composition should consider Web services having the functional, behavioral, semantic, and non-functional properties. Composing Web services by taking into account their conformance from functional, non-functional, behavioral, and semantic aspects together is a challenging issue (Papazoglou and Van Den Heuvel). However,consideration of these aspects together increases the performance of the solution with increased level of user satisfaction and higher degree of automation.

1.3 Objectives

Following objectives have been identified for discovery, selection and composition problem using Heterogeneous Web services.

- To develop framework, approach and prototype for service discovery, selection and composition for SOAP-based Web services using Semantic Web and non-functional characteristics along with sufficient experimental evaluation.
- To develop framework, approach and prototype for service discovery, selection and composition for RESTful Web services using Linked Data principles and non-functional characteristics along with sufficient experimental evaluation.
- To develop framework,approach and prototype for service discovery, selection and composition for Cloud services using Semantic Web and non-functional characteristics along with sufficient experimental evaluation.

1.4 Scope of the Work

The scope of the work for the service discovery, selection and composition problem is as follows.

- To focus on SOAP-based Web services discovery, selection and composition using Integrated approach based Ontology and non-functional characteristics. To evaluate the performance using standard dataset such as Web Service Challenge(WSC). To develop a use case scenario for Healthcare sector.

- To focus on RESTful Web services discovery, selection and composition using Link Open Data (LOD) and QoS-based approach. To evaluate the performance with publicly open datasets such as Linked Open Data. To develop a use case scenario for Public data and Healthcare sector.

- To focus on Cloud services discovery, selection and composition using Ontology and QoS-based approach. To develop a use-case scenario for Healthcare sector.

1.5 Major Contributions of the Work

The contributions of the work are specified as below.

- **Frameworks and approaches for discovery, selection and composition of Heterogeneous Web services** - The frameworks and approaches for service discovery, selection and composition of Heterogeneous Web services have been proposed, i.e., SOAP-based Web services, RESTful Web services and Cloud services.

- **Prototype Development**- Based on the frameworks and approaches, Prototypes have been developed for Healthcare Information System, Population Information System, Healthcare Recommendation System and Healthcare Decision Making System. Publicly available US census data and healthcare dataset have been used to develop the prototype. Moreover, performance evaluation of these prototypes have been done.

1.6 Organization of Thesis

The rest of the thesis is organized by collection of chapters as below. Chapter 2 covers necessary background information of service discovery, selection and composition for the heterogeneous services, such as SOAP-based Web services, RESTful Web services and Cloud services.

Chapter 3 discusses the literature review of the existing service discovery, selection and composition approaches and solutions for Heterogeneous services, such as SOAP-based Web services, RESTful Web services and Cloud services.

Chapter 4 presents the proposed architecture and approaches for Web service discovery, selection and composition. The related architectures and models are evaluated. Finally, the realization of the architecture is presented in the form of experimental work.

Chapter 5 presents the proposed architecture and approaches for RESTful service discovery, selection and composition. The related architectures and models are evaluated. After that, our architecture, its roles and interactions are presented. Finally, the realization of the architecture is presented in the form of experimental work.

Chapter 6 presents the proposed architecture and approaches for Cloud service discovery, selection and composition. The related architectures and models are evaluated. After that, our architecture, its roles and interactions are presented. Finally, the realization of the architecture is presented in the form of experimental work.

Chapter 7 presents the conclusions with an overview of the contributions. Possible future work is also presented.

Chapter 2

Web services Preliminaries

In this chapter, the preliminary concepts of Heterogeneous Web services are presented. Section 2.1 introduces the Service Oriented Computing; Section 2.2 presents Service Oriented Architecture concepts together with semantic aspects of SOAP-based Web services. The basic concepts of Resource Oriented Architecture, REST and RESTful Web services along with semantic aspects are defined in Section 2.3. A comparison between SOAP and RESTful Web services is given in section 2.4. Section 2.5 covers Cloud computing and Cloud services concepts. Section 2.6 describes the QoS model for Heterogeneous Web services.

2.1 Service Oriented Computing

The Service-Oriented Computing (SOC) paradigm uses services to support the development of rapid, low-cost, inter-operable, evolvable, and massively distributed applications. Services are autonomous, platform independent entities that can be described, published, discovered. They perform functions that range from answering simple requests to executing sophisticated business processes requiring peer-to-peer relationships among service consumers and providers ([Papazoglou et al.),(Papazoglou et al.), (Medjahed and Bouguettaya). SOC uses Service Oriented Architecture (SOA) to represent software components into a set of interactive services (Sheng et al.).

2.2 Service Oriented Architecture (SOA)

Service-Oriented Architecture (SOA) (Bianco, Kotermanski, and Merson) is an architectural paradigm for designing and developing distributed systems. Though a lot of definitions (Bianco, Kotermanski, and Merson), (Bose et al.), (Erl) are available, the core idea of SOA revolves around the notion of service. According to (Bianco, Kotermanski, and Merson), a service has the following common properties as an ideal service.

1. A service is self-contained. A service is highly modular and can be independently deployed.

2. A service is a distributed component. A service is available over the network and accessible through a name or locator other than the absolute network address.

3. A service has a published interface. Users of the service need to see the interface and can be oblivious to implementation details.

4. A service stresses interoperability. Service users and providers can use different implementation languages and platforms.

5. A service is discoverable. A particular directory service allows the service to be registered so users can look up the required service.

6. A service is dynamically bound. A service user does not need to have the service implementation available at build time; the service is located and bound at run-time.

SOA(Service Oriented Architecture) has three main components: registry, service customer, and service provider.

The Figure 2.1 shows the interaction among those three components. The service provider advertises the services in the registry. The customer looks for the service and then consumes the services as mentioned in the contract. SOAP-based Web services is the realization of the service oriented architecture.

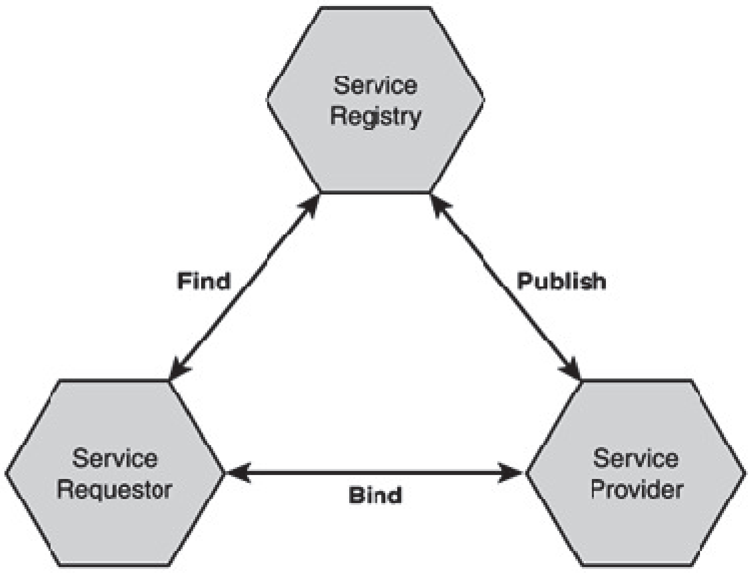


Figure 2.1: Service-oriented architecture

2.2.1 SOAP-based Web services

According to W3C, a Web service is defined as a software system identified by a URI, whose public interfaces and bindings are defined and described using XML language. It's definition can be discovered by other software systems. These systems may then interact with the Web service in a manner prescribed by its definition, using XML based messages conveyed by Internet protocols(Booth et al.).

A Web services architecture model with the three core technologies: (i) WSDL, (ii) UDDI, and (iii)SOAP is shown in Figure 2.2. Among these technologies, UDDI is used as service registry in order to provide a mechanism, where services can be published and retrieved, WSDL is used in order to describe Web services; the service advertisements in the UDDI registry are also based on WSDL. SOAP is used in order to invoke a service and exchange messages between applications. Combined, WSDL, UDDI, and SOAP facilitate the application to realize service-oriented concepts over the Web.

2.2.2 Semantic aspects of SOAP-based Web services

In this section, I define the concepts, which are related to the semantic aspects of the SOAP-based Web services.

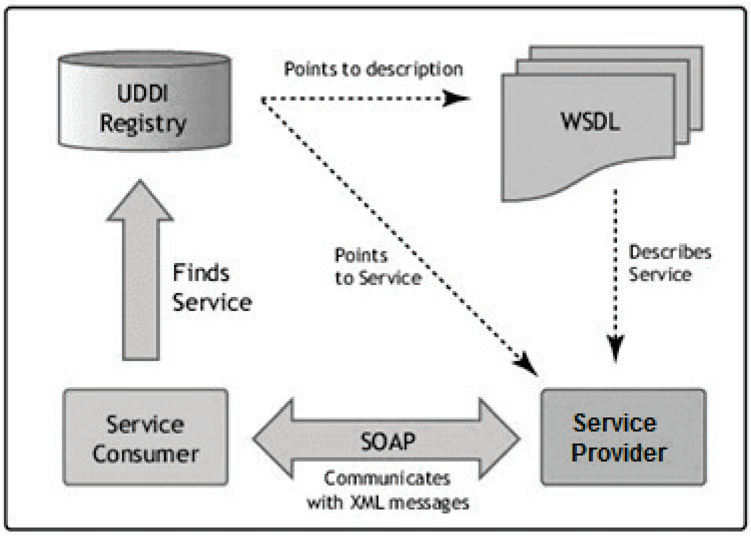


Figure 2.2: Web services architecture model

(I.) **Semantic Web**

Web service technologies, such as WSDL, UDDI and SOAP describe the syntactical aspects of a Web service, providing only a set of rigid services that cannot be adapted to a changing environment without human intervention ([Benatallah et al.). Thus, common standards lack of machine-interpretable information regarding functional and non-functional aspects. This is a major shortcoming of the Web services technologies, the concept of the Semantic Web plays a key role to resolve this issue. Tim Berners-Lee has proposed the idea of the Semantic Web (Berners-Lee, Hendler, Lassila, et al.). Lee has defined the Semantic Web as: ”The Semantic Web is an extension of the current Web, in which information is given well-defined meaning, better enabling computers and people to work in cooperation. It is the idea of having data on the Web defined and linked in a way that it can be used for more effective discovery, integration and reuse across various applications”. ”The Semantic Web is an evolving extension of the World Wide Web in which, Web content can be expressed not only in natural language, but also in a format that can be read and used by computers, thus permitting a more automated and effective way to find, share, and integrate information. Semantic Web Technologies (SWTs) facilitate exchange of information among various applications in a meaningful way by providing precise and unique meaning and context to the content and easing interaction between the human users and a computer system, or between two computer applications (Unhelkar and Murugesan),(Ummel). Semantic Web technology, as shown in Figure 2.3 is developed as layered cake model.

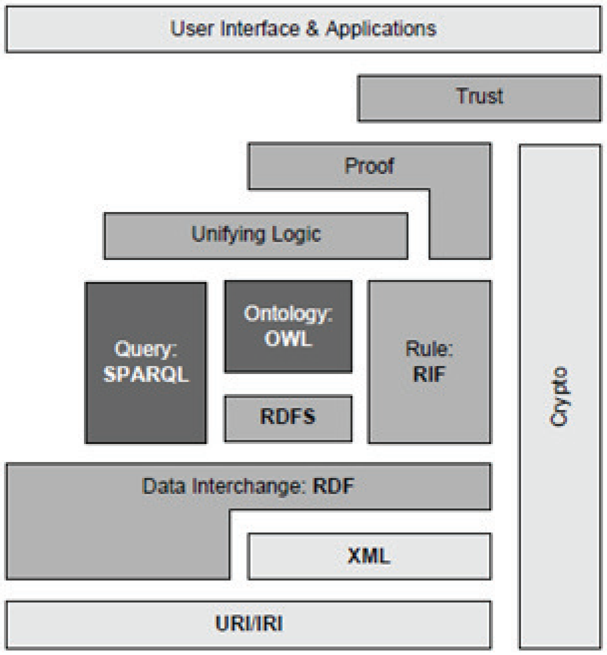


Figure 2.3: Semantic Web Reference Architecture Version 4

At the base of the layer is an URI. The URI and XML schema are the foundation of the Semantic Web architecture. XML (Bray et al.) supports users to add arbitrary structure to their documents by creating tags to annotate a web page. Although the meaning of XML tags is intuitively clear, tag names by themselves do not provide semantics. RDF (Resource Description Framework)(Lassila, Swick, et al.) and RDFS (Brickley and Guha) provide a basic framework for expressing meta-data on the Web, while current developments in web-based knowledge representation, such as OWL (McGuinness, Van Harmelen, et al.) build on RDF to provide more sophisticated knowledge representation support. Logic layer enables intelligent reasoning with meaningful data.

(II.) **Ontology**

In the field of Semantic Web, ontologies are considered as the backbone for machine understandable data and allows exchanging information more meaningfully for humans and computers alike. Ontologies can be described in many different formats as a result, there is no commonly agreed definition of the term ontology (Chandrasekaran, Josephson, and Benjamins). However, the Grubers quote is widely accepted as a common definition of ontology from a technical point of view (Gruber):ontology is a formal, explicit specification of a shared conceptualization". This definition highlights distinct features ontology needs to address: First of all, it is *formally* specified, i.e., ontology makes use of a defined ontology language. Second, *conceptualization* refers to an abstraction of a domain which includes the relevant concepts in that domain. Third, ontology is based on shared knowledge, i.e., it represents an agreed viewpoint.

A more comprehensive description of ontology from computer science perspective is provided by (Lacy): Computer science ontologies serve a similar function as database schema by providing machine-understandable semantics of information sources through collections of terms and relationships. The semantics support a shared and common understanding of a domain, that can be communicated between people and software".

(a.) **Foundational and Domain ontologies**

Based on the application domain or generic model, ontologies can be defined within two very different perspectives: (i)Foundational ontologies, also known as upper or top-level, are a model of the common objects applicable across a wide range of domains and developed to represent explicitly a viewpoint of a reality. They are built upon a core vocabulary that contains the terms and associated object descriptions, as they are used in various relevant domain sets. The very common foundational ontologies are BFO,GFO, SUMO and DOLCE, amongst others (Keet). (ii) Domain ontologies, describe a set of representational primitives that model a domain of knowledge or discourse, providing a common and unambiguous understanding of a domain for both the users and the system ([Zhang et al.). It models a specific domain, without any pretension of generality.

2.2.3 Semantic description of SOAP-based Web services

The Ontologies are expected to play a central role to empower Web services with expressive and computer interpretable semantics. The combination of these powerful concepts (i.e., Web service and ontology) has resulted in the emergence of a new generation of Web services called Semantic Web services (McIlraith, Son, and Zeng),(Martin et al.),(Miller et al.),(Medjahed and Bouguettaya). Semantic Web services provide an open, extensible, semantic framework for describing and publishing semantic content, improved interoperability, automated service composition, discovery and invocation, access to knowledge on the Internet (McIlraith, Son, and Zeng). In order to define the meaning of distinct service components by semantic annotations or enhancements of a service description, it is necessary to have a domain model which can be used as a knowledge base. Most probably, the best-known knowledge base format is ontologies.

Most approaches intend to describe the semantics of Web services, either with novel semantic description languages (Martin et al.), (Arroyo, Stollberg, and Ding) or by extending the syntactic mechanisms (Miller et al.), (Lausen and Farrell) using domain ontologies to annotate data and Web services.

(I.) **OWL-S**

OWL-S (Martin et al.) is an ontology based on Web Ontology Language (OWL) for developing Semantic Web services by annotating syntactic description formats, such as WSDL. OWL-S comprise of three main components: the service profile, the process model, and the service grounding, which define what the services does, how the services works, and how to access the service, respectively. OWL-S focuses on isolating the grounding and abstracting views, when describing the data associated with Web services. Abstract view binds the data to an OWL conceptual description while Grounding view specifies the low-level representation of data by following XML Schema (Biron and Malhotra).

(II.) **WSMF and WSMO**

WSMF (Fensel and Bussler) offers the description and development of Semantic Web services with a conceptual model which focuses on isolation between Web services. WSMO (Arroyo, Stollberg, and Ding) is a language and ontology based on the WSMF conceptual model that represents various aspects of Semantic Web services.

(III.) **WSDL-S**

WSDL-S, (Miller et al.) offers annotation of WSDL with some extensions associated with operations and messages. These extensions identify the concepts of domain models in order to specify the semantics of messages as well as the preconditions and effects of operations. WSDL-S is also considered as a lightweight approach for semantic annotation.

(IV.) **SAWSDL**

SAWSDL defines a set of extension attributes to WSDL 2.0 (with WSDL 1.1 support) in order to represent the semantics of WSDL (Lausen and Farrell). The purpose SAWSDL is to define how semantic annotation of WSDL is accomplished. It only offers the means to bind ontology concepts to WSDL annotations.

2.2.4 Service Discovery, Selection and Composition for SOAP-based Web services

Following section describes the basic concepts of Service discovery, selection and composition of SOAP-based Web services.

(a.) **Service Discovery**

Web service discovery is a process of finding most suitable service from the repository according to requesters' requirement. (Singh and Huhns). Web services discovery approaches are classified into four main categories (Zunino and Campo),(Mukhopadhyay and Chougule): (i)*Information Retrieval-based approaches*, (ii)*Semantic Web-based approaches*, (iii) *QoS-aware approaches*, and (iv) *Context-based approaches*.

By adapting existing Information Retrieval (IR) techniques, some researchers have proposed to treat descriptions of Web services as documents to reduce the problem of discovering relevant services. Semantic Web-based approaches propose to annotate the service descriptions with meta-data, such as concept definitions from shared ontologies or sometimes referred as semantics, which gave the notion of Semantic Web Services. The Semantic Web approaches depend on shared ontologies and annotated resources, whereas IR-based ones depend on textual descriptions.

By definition, the context is a situation of an entity (person, place or object) that is relevant to the interaction between a user and an application (Newcomer). Therefore, considering the context in the query-service matching process can improve the quality of the retrieved results. However, contextual information is highly interrelated and has many alternative representations (Pokraev et al.),(Broens et al.). This makes it difficult to interpret and use.

The QoS is a set of non-functional attributes that may affects the quality of the service provided by a Web service. QoS parameters describe non-functional aspects of Web services and they are used to evaluate the degree that a Web service meets specified quality requirements in a request. QoS-aware service discovery provides quality guarantee with increased level of satisfaction to the user.

(b.) **Service Selection**

Service selection is a very complex and challenging task, especially if it takes a variety of different non-functional properties into account. The rapid growth in the number of services increased the importance of the service selection task due to the presence of low quality services. In the state of the art approaches for service selection, in order to filter out low quality candidates during the selection process, non-functional aspects are exploited as the key decision making criteria. As a result, quality of service (QoS) is a significant concept since QoS properties describe non-functional aspects of Web services and evaluate their conformance degree. The Web Services Selection process is broadly classified into three main approaches (Sathya et al.): (i) *Functional- based approach*, (ii)*Nonfunctional- based approach*, and (iii) *User- based approach*.

The functional-based service selection approach represents the Static and Dynamic semantics. Selecting an appropriate service is concerned with retrieving functional descriptions from service repositories and then ensuring that the described and required interfaces match with each other. Static semantics represents the properties of messages and operation semantics. Dynamic semantics represents the properties of behavior and operation logic. With the rapidly growing number of available services, customers are presented with a choice of functionally similar services. This choice allows customer to select services that match other criteria, often referred to as non-functional attributes. The non-functional-based service selection represents the QoS and Context in Semantic Web service selection. The properties of QoS may be (security, reliability, response time, cost etc.), the properties of context may include context of customer (location, customerś name) and context of service (providerś details, service descriptions etc,). User based approach represents the selection of best service among numerous discovered services based on customer's feedback, trust and reputation.

QoS-based approaches can be grouped in two major categories (Alrifai and Risse): (i) the multi-objective and (ii) the mono-objective optimization. The first one can utilize a global selection method (Zeng et al.), ([Zeng et al.) or a local selection method (Benatallah et al.) or a hybrid selection method (Alrifai and Risse). The global selection method can offer the optimal solution with an exponential complexity; however the local method has only a linear complexity, but cannot deal with the global constraints. The third category is a combination of the two approaches, it has a reduced complexity in comparison with the global approach, and able to deal with the global constraints (Fethallah et al.).

(c.) **Service Composition**

Web services composition is a process to combine more than one service to create composite service (Dustdar and Schreiner), (Rao and Su), (Alamri, Eid, and El Saddik). The service composition support the users to create applications on top of the native service description, discovery, and communication capabilities of service oriented computing. Service composition can be either performed by composing elementary or composite services. Composite services, in turn are recursively defined as an aggregation of elementary and composite services. When composing Web services, the business logic is performed by several services. It is identical to workflow management, where the application logic is realized by composing autonomous applications. A client invoking a composite service can itself be exposed as a Web service. Some of the service composition solutions (de Oliveira Jr and de Oliveira) identify the need of QoS attributes to get the optimum solution. Service composition methods are classified based on features, such as composition pattern, composition handling, and composition time by several authors (Sheng et al.),(Dustdar and Schreiner),([Li et al.),(Cardoso, Sheth, and Yu),(Rao and Su). The service composition approaches mainly classified into two categories (Liu, Cui, and Gu),(Laliwala et al.): (i) *workflow-based*, and (ii) *semantic- based* or *artificial intelligence (AI) planning-based*.

Composition based on workflow is performed by defining a Web service execution process, in which the control-flow and the date-flow are explicitly specified among those Web services. BPEL4WS or WS-BPEL (Standard) is a process execution language that combines other standards of Web services composition, such as WSFL (Leymann et al.) from IBM, which is graphics-oriented, and XLANG (Thatte) from Microsoft, which is well structured. Others like WSCI (Arkin et al.), ebXML (Gibb and Damodaran), BPML (Thiagarajan et al.), XPDL (Shapiro) and WSMF (Fensel and Bussler) are all composition standards developed recently, which are mainly based on workflow.

Based on the composition pattern, service composition process is divided into two categories: service orchestration and service choreography (Peltz). Service orchestration represents a business process which coordinates and interacts among the various services, by describing an invocation order of Web services. The standard for Web services orchestration is Web Services Business Process Execution Language (WS-

BPEL), which is widely accepted by the industry. Service Choreography describes collaboration between Web services that focuses on peer-to-peer interaction, where all participating Web services work equally and do not rely on central coordinator. The choreography mechanism is supported by the standard WS-CDL(Web Services Choreography Description Language) (Kavantzas et al.).

Based on the composition time, service composition could be categorized into static or dynamic type. Static composition performs the integration of services at design time. Static composition works fine if the business entities participating in the process are relatively unchanged, and service functionalities or composition requirements do not, or rarely change. Static composition is not flexible in conditions when there are frequent runtime modifications of requirements or services that cannot be considered at design time. In contrast, a dynamic composition offers to determine and replace services at runtime. Dynamic composition requires the execution environment to support automatic discovery, selection, and binding of service components (Sheng et al.).

Based on the automation feature, services composition can be defined into three categories: manual, automated, and semi-automated (Milanovic and Malek). In manual approach, a service provider generates an abstract composition plan, using standard language, such as WS-BPEL. Then, the user binds the Web services to the abstract process manually. The manual composition is a time-consuming and error-prone process.

Automated services composition can be classified into the Semantic Web and Artificial Intelligence (AI) planning techniques: The Semantic Web allows the representation and exchange of information in a meaningful way, facilitating automated processing of descriptions on the Web. Annotations on the Semantic Web express links between information resources on the Web and connect information resources to formal terminologies. These connective structures are called ontologies, which are a widely accepted state-of-the-art knowledge representation. The extensive usage of ontologies allows semantically enhanced information processing and support for interoperability. AI planning problem can be described as a tuple $\langle$ S, S0, G, A, T $\rangle$ (Rao and Su), where

S is the set of all possible states of the world; S0 denotes the initial state of the world; G denotes the goal state of the world the planning system attempts to reach; A is the set of actions the planner can perform in attempt to reach a desire goal, and The translation relation T= SxAxS defines the precondition and effects for the execution of each action. There are some shortcomings for using AI planning for Web service composition. A mapping of the states and actions of AI planning problems to the operations of Web services is necessary for Web service composition. One of the basic concepts of Web services is the independence of the interface from the internal processing. A Web service only has to behave according to the specified interface, but the caller doesnt know about the internal states behind the Web service interface. A planner can only try to derive the state depending on the exchanged messages. This leads to the problems of partial observability of state and ambiguity of state description.

By providing a collection of atomic or composite services and a users request, a service composition can be created automatically. However, achieving a fully automated services composition is very difficult and shows several open issues (Medjahed, Bouguettaya, and Elmagarmid).

2.3 Resource Oriented Architecture (ROA)

The ROA (Fethallah et al.) provides guidelines to implement the REST-style architecture. ROA specifies four principles: (i) Resources. (ii) Their names (URIs) (iii) Their representations, &(iv) The links between them and four properties: (i) Addressability. Addressable entities expose a URI for every piece of information they might serve. (ii) Statelessness. It means every HTTP request occurs in complete isolation without depending on information from previous requests. (iii) Connectedness. A Web service is connected to the extent that you can put the service in different states just by following links and filling out forms. (iv) A uniform interface. HTTP is an uniform interface. Probably HTTP methods are not a perfect interface but what is important is the uniformity (Fethallah et al.).

2.3.1 REpresentational State Transfer (REST)

The REST was first described in Fieldingś PhD thesis (Fielding). REST (Fielding) defines the behavior of web application communication, where application presented with a network of web pages (a virtual state-machine), the user progresses through an application by selecting links (state transitions), resulting in the next page (representing the next state of the application) being transferred to the user. The main concept in REST concerns the resource. REST architectural style is not closely attached to HTTP (Fielding et al.), even though HTTP is commonly adopted for its implementation. The key principles of REST are the following:

1. Use URIs for resource identification. The resources are represented by servers using URIs, the clients invoke for interaction (Paganelli et al.).

2. Adoption of a uniform interface. The conversation with the resource is totally presented with four primitives: create, read, update and delete. In HTTP, they are handled by the PUT, GET, POST and DELETE verbs respectively (Pautasso, Zimmermann, and Leymann).

3. Adoption of self-descriptive messages. Every message has the information needed for its management. Meta-data is used for data negotiation, errors alertness, etc.

4. Adoption of stateless interactions. Each request from client to server must contain all of the information necessary to know the request. Session state is maintained by the client. The server is responsible to manage and store the state of the resources it exposes.

Representational State Transfer (REST) has gained widespread acceptance across the Web as a simpler alternative to SOAP- and Web Services Description Language (WSDL)-based Web services. Key evidence of this shift in interface design is the adoption of REST by mainstream Web 2.0 service providers including, Yahoo, Google, and Facebook, who have deprecated or passed on SOAP and WSDL-based interfaces in favor of an easier-to-use, resource-oriented model to expose their services.

2.3.2 RESTful Web services

RESTful Web services (Richardson and Ruby) follow traditional mechanism of World Wide Web (WWW), which is based on an REST architectural style. It consists of service requester and service provider as depicted in Figure 2.4. Representations of resources, which are identified by URI (Uniform Resource Identifier) are transferred using request and response messages. During the communication, semantics of data is defined implicitly.

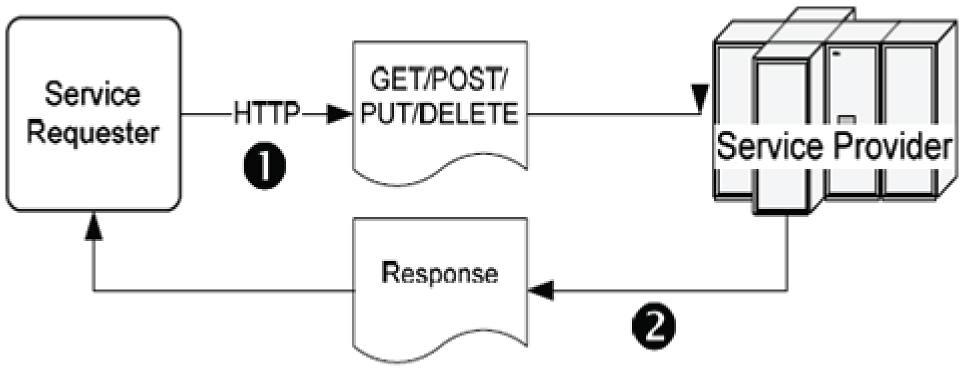


Figure 2.4: RESTful Web Services Architecture

RESTful Web services are based on HTTP protocol and its methods, such as PUT, GET, POST, and DELETE. These Web services are better integrated with HTTP than SOAP-based services and they do not require XML SOAP messages or WSDL service definitions.

2.3.3 Semantic aspects of RESTful Web services

In this section, the concepts which provide semantic annotation to the RESTful-based Web services, such as Linked Data, Linked Data Services and RDF are defined as follows.

(I.) **Linked Data**

The Linked Data provides a set of best practices for publishing and connecting structured data over the Web as an upcoming alternative solution of the Semantic Web. Linked Data is an effort to create a web of data, parallel to the web of documents -

the Web, we know and use today (Berners-Lee, Bizer, and Heath).Complex information may be built aggregating simpler information units, but unlike the current Web paradigm, which conceives complex information as a whole, the information units are individually addressable and linkable (Paganelli et al.). Linked Data describes a method of publishing structured data so that it can be interlinked. It shares information over HTTP as RDF and enables querying over data. Figure 2.5 shows the evolution of Web and position of Linked Data.

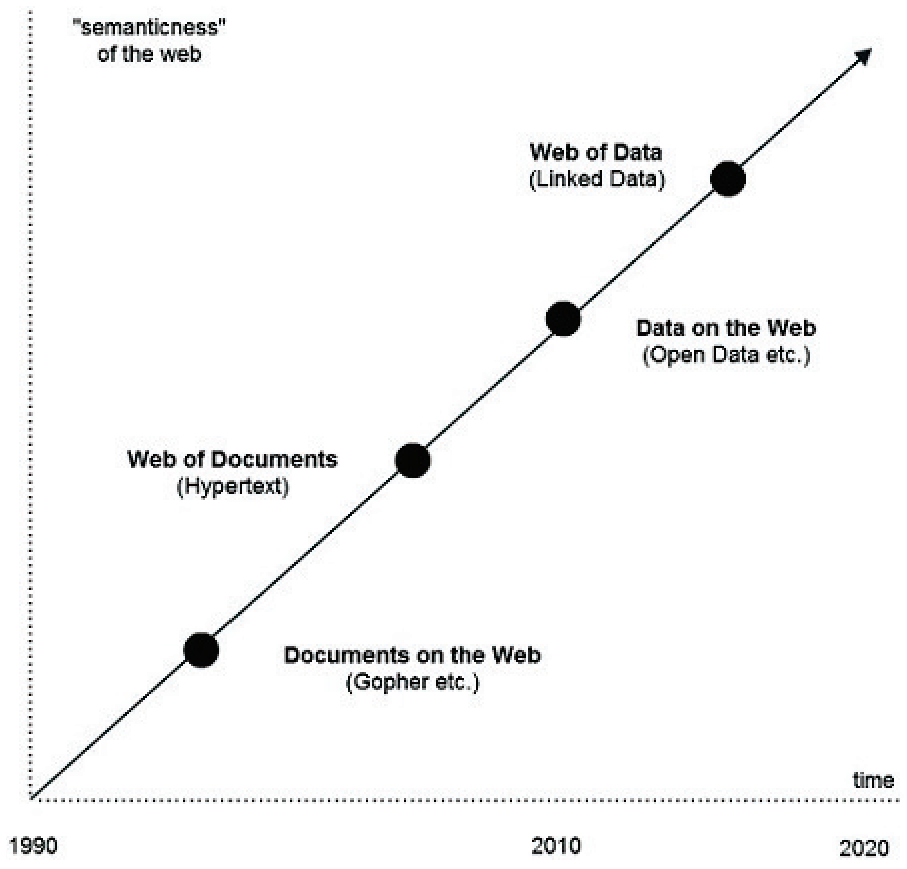


Figure 2.5: Evolution of the Web (Bauer and Kaltenböck)

In 2006, Timbers Lee published the Linked Data principles (Berners-Lee, Bizer, and Heath):

1. Use URI (Berners-Lee, Fielding, and Masinter) as names for things.
2. Use HTTP URIs, so that people can look up those names.

3. When someone looks up a URI, provide useful information, using standards (RDF, SPARQL (PrudH́ommeaux, Seaborne, et al.)).
4. Include links to other URIs, so that they can discover more things (Berners-Lee, Bizer, and Heath).

The Linked Data paradigm represents a global browsable information space, where data from various sources are connected and integrated to create new information, offering new possibilities for domain-specific applications (Berners-Lee, Bizer, and Heath).

(a.) **Linked Data Services**

Linked Data Services provide a Linked Data interface for data services. To make these services adhere to Linked Data principles a number of requirements have to be fulfilled: (i)the input for a service invocation with given parameter bindings must be identified by a URI; (ii) resolving that URI must return a description of the input entity, relating it to the service output data; (iii) the description must be returned in RDF format. Ssuch services are called as Linked Data Services (LIDS)([Speiser and Harth).

A LIDS provides HTTP URIs for entities representing service inputs that encode parameters as key-value pairs in the query string. Dereferencing the URI via HTTP GET returns an RDF description of the service input entity, it's relation to the service output and the output data itself. Both input and output of LIDS are formally described using SPARQL (Speiser and Harth).

(II.) **Resource Description Framework (RDF)**

According to the REST principles, the resourceś uniform interface is exclusively used to exchange representations. As stated above, the diversity of those representations is the key to differentiating resources and their behaviors. Therefore, a data model that is flexible enough to support those multiple kinds of representations is needed. RDF is such a data model. Being specifically designed for the Web, it uses

URIs to name things, which is consistent with the REST and linked data principles. It is based on graphs that allow representing a wide range of data structures.

RDF is a W3C specification for representing semantic information on the Web. It is ideal for representing meta-data about Web resources. RDF information is processed by software applications rather than humans. It provides a common platform for exchanging of information between applications (Manola, Miller, and McBride).

RDF is based on identifying resources through URI using properties and property values. The property values along with the resources represent RDF graphs. A RDF graph is a collection of nodes and arcs, where a node represents a subject or object while an arc represents a predicate. A subject and an object is an individual like, John or a thing like chair. In terms of English grammar, a subject or object is a noun or pronoun, a predicate shows the relationship between a subject and an object; grammatically a predicate is a verb describing an action or state (Manola, Miller, and McBride).

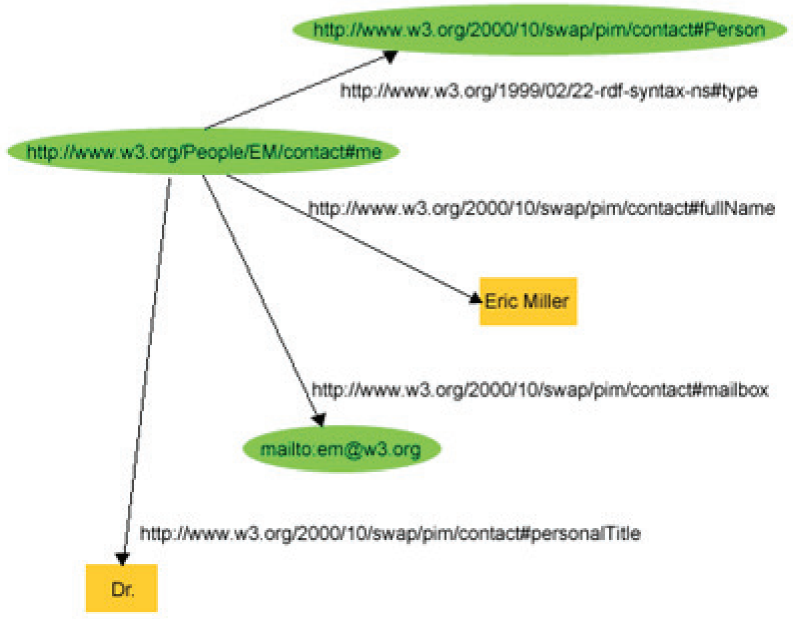

http://www.w3.org/2000/10/swap/pim/contact#Person
http://www.w3.org/1999/02/22-rdf-syntax-ns#type
http://www.w3.org/People/EM/contact#me
http://www.w3.org/2000/10/swap/pim/contact#fullName
Eric Miller
http://www.w3.org/2000/10/swap/pim/contact#mailbox
mailto:em@w3.org
http://www.w3.org/2000/10/swap/pim/contact#personalTitle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

Figure 2.6: An RDF Graph Describing Eric Miller (Manola, Miller, and McBride)

A RDF graph describing Eric miller is shown in Figure 2.6. The graph shows four sets of linked information: the first being me is a type-of person, second being my personal title is Dr, third being my mailbox is em@wm.org and the fourth being my full name is Eric Miller. This graph describes Dr Eric Miller is a person who can be reached at em@wm.org. As RDF is semantic, a software application would actually understand this statement and could share it or use it. A RDF graph is written down in the form of RDF triples, where each statement in the graph is a triple (Manola, Miller, and McBride).

(III.) **SPARQL Protocol and RDF Query Language (SPARQL)**

SPARQL (PrudH́ommeaux, Seaborne, et al.) is a query language for querying RDF graphs. It was designed to meet the use cases and requirements identified by RDF. SPARQL syntax is similar to that of the widely used Structured Query Language (SQL). A typical SPARQL query consists of prefix, select, update and where clauses. In the below example all the entries are being selected from the given URL.

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

The prefix clause is used for abbreviating URIs in above example dc will be used instead of the given URL. The select clause is used for retrieving data, while update clause is used for updating. In the following example entries from the title column are being selected. ”WHERE clause is used for specifying the conditions of selection.

2.3.4 Semantic Description of RESTful Web services

Several efforts have been done to develop mechanisms to describe semantic description of the RESTful Web services, such as hRESTS and SA-REST as follows.

(i.) **hRESTS (HTML for RESTful Services)**

hRESTS (Kopecky, Gomadam, and Vitvar) is a microformat that provides machine-readable representation with semantic annotations for available RESTful Web services and Web APIs. The microformat takes the benefit of XHTML code to annotate different aspects of the associated services. It defines the main features of services, such as services, inputs and outputs. hRESTS is designed for RESTful Web services, which avoids unnecessary complexities faced with existing one. The required efforts from the developer are also reduced since a separate description of the service is no longer needed.

(ii.) **SA-REST (Semantic Annotations for REST)**

Similar to hRESTS, SA-REST (Gomadam, Ranabahu, and Sheth) is also offers techniques to add semantics to RESTful Web services. It uses the RDFa syntax to describe services, operations. Same advantages and disadvantages are presented by SA-REST in comparison with hRESTS format. Moreover, since SA-REST is strictly based on RDF concepts, the developer required the knowledge of RDF.

2.3.5 Service Discovery, Selection and Composition for RESTful Web services

This section describes the basic concepts of Service discovery, selection and composition of RESTful Web services.

(a.) **Service Discovery**

Service discovery represents the process to discover similar types of services in the same domain and to link to them. Service discovery supports users to search services that could provide data needed to access another service or could consume data received from another service. Available service discovery solutions use a directory-based approach, which is suitable for SOAP-based Web services. For RESTful services, a peer-to-peer discovery mechanism is required that works without any dependence on a central directory. The process works by identifying the different links as it comes across new resources and generating a graph connecting them.

(b.) **Service Selection**

Web service selection is the process to identify the best candidate services from a set of services with similar functionalities, but having different values of Quality of Service (QoS). Nowadays, researchers have used QoS parameters for selection process based on Multi-Criteria Decision Making (MCDM) techniques. The approaches utilized for Web service selection are explored based on following set of characteristics:

- **QoS:** This represents the approaches that utilize QoS as the parameter to take decision for selection.
- **User Preference:** This represents the approaches that consider user preferences in order to take into account the priority of service requesters.
- **Scalability:** This represents the approaches, which consider numerous characteristics and ranking processes that occur concurrently with accuracy of the results.
- **Automatic:** This represents the selection of service from the available services in a transparent way, without involving, bothering, or distracting its user.

(c.) **Service Composition**

Service composition is made possible by using the filtered resources provided through selection. A graph is constructed starting from a resource and then traversing the parent, consumer and producer links recursively. At each page, the descriptions are extracted and converted to RDF to update the graph. This way, a software program that doesnt have the search parameter to access a specific resource could traverse the graph to figure out what other information could be used to present the solution to the user.

2.4 Comparison between SOAP-based Web services and RESTful Web services

A detailed comparison between two Web services is derived from (Pautasso, Zimmermann, and Leymann),([Li et al.),(Liu),(AlShahwan, Moessner, and Carrez) as shown in Table 2.1.

Table 2.1: Comparison of Web services technologies

Features	SOAP-based WS	RESTful WS
Architectural model	SOA	REST
Commu. protocol	TCP, FTP, HTTP	HTTP only
Security	WS-security specification	HTTP Security
Service identification	URI, WS-Addressing	URI
Service description	WSDL	WADL
Service discovery	UDDI	No centralized registry
Service Composition	BPEL WS-CDL, User defined	Mashup, User defined

2.5 Cloud Computing

Cloud computing has been coined as an umbrella term to describe a category of sophisticated on-demand computing services initially offered by commercial providers ([Ardagna and Pernici). It denotes a model on which a computing infrastructure is viewed as a Cloud, from which businesses and individuals access applications from anywhere in the world on demand (Buyya et al.). The main principle behind this model is offering computing, storage, and software as a service.

The US National Institute of Standards and Technology (NIST) has defined cloud computing as Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction". This cloud model is composed of five essential characteristics, three service models, and four deployment models (Mell and Grance) as follows.

- **Five characteristics:** on-demand self-service, broad network access, resource pooling, rapid elasticity, and measured service.
- **Four deployment models:** private Clouds, community Clouds, public Clouds, and hybrid Clouds.
- **Three service models:** Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS).

IaaS Clouds offer computing resources, such as processing power, storage, networks, and other fundamental computing resources. The underlying Cloud infrastructure is managed by a provider. However, users have the flexibility to select their virtual machine images and to deploy these applications. In the PaaS model, providers supply clients with tools and services to develop software applications. In addition to the IaaS restrictions, PaaS users do not have the ability to manage or control their virtual machine images and servers. SaaS providers allow customers to use the applications such as Google Docs and Microsoft office Web Apps running on a Cloud infrastructure.

Moreover, Clouds deployment model can be classified as private, community, public, or hybrid Clouds. A Private cloud is utilized by an organization and is neither shared with other organizations nor with the general public. In contrast, a public clouds services are accessible to the general public and in the case of community Cloud, the infrastructure is shared by a number of organizations. A Hybrid Cloud offers services deployed on two or more Clouds.

2.5.1 Cloud Services

Cloud computing permits a service-provisioning model, which typically involves the provisioning over the Internet of dynamically scalable and virtualized services (Badidi). Applications or services offered by means of cloud computing are called Cloud services. Cloud services are defined by (Soman) as: Cloud Services are Information Technology services that satisfy the following criteria:

1. Consumers neither own the hardware on which data processing and storage happens, nor the software that performs the data processing.

2. Consumers have the ability to access and use the service at any time over the Internet.

In fact, the cloud computing paradigm is more or less based on the principles of SOC, in particular the SaaS service model. Advances in SOC can benefit Cloud Computing in several ways through Cloud services (Dillon, Wu, and Chang).

- **Service Description for Cloud Services:** WSDL and WADL are two widely used interface languages to describe Web services as defined earlier. They have been utilized to describe Cloud API specification.

- **Service Discovery for Cloud Services:** Various service discovery models can be leveraged for cloud resource discovery, selection and service-level agreement verification.

- **Service Composition for Cloud Service:** Since Web services are born to compose business applications, a great deal of research in this area can be leveraged for cloud services integration, collaboration, and composition.

- **Service Management for Cloud Service:** Research and practices in SOA governance and services management can be adapted and reused in the cloud infrastructure management.

Cloud platform delivers Cloud services as the products and solutions of it. Cloud services can be represented by their static and dynamic features. Static features are represented at deployment time. Common static features are resources (CPU, memory, software) and dependences between resources. Dynamic features are represented at execution time. Common dynamic features are QoS rules and the price of resources (García and Blanquer).

2.5.2 Service Discovery, Selection and Composition for Cloud services

Following section describes the basic concepts of Service discovery, selection and composition of Cloud services.

(a.) **Service Discovery**

Service discovery is a procedure of searching for required services, in which their functional and non-functional semantics satisfy a customers goal. Cloud services are typically accessed using brokers. The broker allows customers to submit a service request to Cloud request, including required set Service Level Agreement (SLA) objectives for that service. The broker will then proceed to match available service descriptions of Cloud provider to service request description and find candidate services, which can provide expected functionality. However, only functional service descriptions are not sufficient for service discovery process (Liu et al.).

(b.) **Service Selection**

Similar to SOAP-based and RESTful Web services, Cloud service selection is the process of selecting a service that fulfills the given user requirements from a list of discovered services. The selection process is become more difficult when it is performed for composite services, where both functional and non-functional requirements should be take into account by the selection process (Jrad et al.).

(c.) **Service Composition** In the service composition process, a broker combines a set of services from multiple providers, and delivers the combined service as a single virtualized service to a consumer. During the composition of services, not only service functionality is utilized when selecting services to generate composition. Besides the fulfillment of SLAs (Service-Level Agreements) (Bose et al.), it is important to take into account metadata about the services, such as QoS, prices, etc., because multiple equivalent services offered by the set of cloud platforms that are being integrated (Cavalcante et al.).

2.6 QoS Model for Heterogeneous Web services

The QoS for a service can be represented through non-functional characteristics with quantitative parameters. It defines the various non-functional parameters, such as throughput, response time, availability, reliability cost, security etc (de Oliveira Jr and de Oliveira), (Al-Masri and Mahmoud), (Vu, Hauswirth, and Aberer), (Mallayya, Ramachandran, and

Viswanathan), (Rajeswari et al.), (Alrifai, Risse, and Nejdl). I have considered QoS-based approach by incorporating non-functional parameters, such as Throughput and Response Time, Cost and Availability in the service discovery selection and composition process for the Heterogeneous Web services. A brief introduction of the QoS model is presented in this section.

I. **Throughput**

The throughput (Yang, Zhang, and Lan),(Al-Masri and Mahmoud) refers to the number of service requests R that can be processed by a service s within a given period of time. A services throughput $Q_{TH}(s)$ can be expressed by equation (2.1) as shown below.

$$Q_{TH}(s) = \frac{\#R}{Time - period} \tag{2.1}$$

Where $\#R$ is the number of service request. Depending upon the services characteristics, the period of time may vary from millisecond (ms) to minute.

II. **Response Time**

The response time (Yang, Zhang, and Lan),(Al-Masri and Mahmoud) refers to the time taken to send a request and receive a response through service execution. A services response time $Q\_RT(s)$ can be expressed by equation (2.2) as shown below.

$$Q_{RT}(s) = ET + WT \tag{2.2}$$

Where, ET is the execution time of service s and WT is the waiting time of service s. The formulas to calculate aggregate values of response time and throughput for sequential execution pattern are presented in Table 2.2,which are inspired by (Mabrouk et al.). In serial execution pattern, services are executed one after another and no overlap is considered between execution periods of Web services.

Where $Q_{RT}(si)$ is the Response Time of a service si and $Q_{TH}(si)$ is the Throughput of a service si. The QoS vector of ith composite service is calculated by equation (2.3) as follows.

$$Q(S) = Q_{RT}(S) + Q_{TH}(S) \tag{2.3}$$

Table 2.2: QoS aggregation function

Sr. No.	**QoS Parameters**	**Aggregation Function**
1	Response Time	$Q_{RT}(s) = \sum_{i=1}^{n} Q_{RT}(si)$
2	Throughput	$Q_{TH}(s) = \min_{i=1}^{n} Q_{TH}(si)$

2.6.1 Calculation of overall QoS Score

The Overall QoS score can be obtained by following the Simple Additive Weighting (SAW) technique proposed by (Yoon and Hwang)and used by (Vu, Hauswirth, and Aberer),(Ouzzani and Bouguettaya),([Zeng et al.) to select an optimal Web service using local optimization technique. There are two main phases to apply SAW method: scaling and weighting which are defined as follows.

(I.) **Scaling:**

The QoS parameters could be either positive or negative, thus some QoS values need to be maximized, (i.e., the higher the value, the higher the quality), whereas other values have to be minimized, (i.e., the higher the value, the lower the quality). To perform this, the scaling phase normalizes each QoS parameter value according to the following formulas. Scaling could be categorized into Positive Scaling and Negative Scaling.

(a.) **Positive Scaling:** The objective of this scaling is that we want to maximize the value of QoS criteria. It defines the scaling for positive criteria (i.e. when the higher value the higher the quality) e.g.: Throughput.

(b.) **Negative Scaling:** The objective of this scaling is that we want to minimize the value of QoS criteria. It defines the scaling for positive criteria(i.e. when the higher value the lower the quality) e.g.: Execution Time.

The equations 2.4 and 2.5 defines the positive and negative scaling values of candidate service (CS) as follows.

$$CS_p = (q - q_{min})/(q_{max} - q_{min}), if qmax - qmin \neq 0 \tag{2.4}$$

$$CS_n = (q_{max} - q)/(q_{max} - q_{min}), if qmax - qmin \neq 0 \tag{2.5}$$

Where, q is QoS value of respective service, CSp is positive scaling, CSn is Negative scaling, q_{min} is minimum QoS value of respective criteria associated with service, q_{max} is maximum QoS value of respective criteria associated with service.

The authors (de Oliveira Jr and de Oliveira) have proposed the equations 2.6 and 2.7, which provides a function to calculate scaled value of a criterion q of candidate services CS considering if q is positive means higher the value the higher the quality or negative means the higher the value the lower the quality. In short, scaled value is determined by considering the highest and lowest values of a given criterion of the candidate services in the candidate list.

$$s(CS, q) = \begin{cases} \frac{overall(CS,q) - q_m(q)}{q_M(q) - q_m(q)} & if q_M(q) - q_m(q) \neq 0 \\ 1 & otherwise \end{cases} \tag{2.6}$$

$$s(CS, q) = \begin{cases} \frac{q_m(q) - overall(CS,q)}{q_M(q) - q_m(q)} & if q_M(q) - q_m(q) \neq 0 \\ 1 & otherwise \end{cases} \tag{2.7}$$

Where, q_M = Maximum $overall(CS, q) : CS \in X$ and qm = Minimum $overall(CS, q) : CS \in X$, X is the list of candidate selection and $overall(CS, q)$ is the quality calculation for the selection service CS and criterion q.

(II.) **Weighting:** This phase computes the overall score taking into account all the involved criteria and the weight assigned to each one of them. The overall QoS score of a single service can be defined by an equation 2.8, which is proposed by (de Oliveira Jr and de Oliveira) as follows.

$$overallQoS(S_{qos/R_{qes}} = \sum_{i=0}^{n}(CS_p/CS_n)) * W \tag{2.8}$$

Where, $OverallQoS(S_{qos})$ is the overall QoS score of all Criteria of a single service, $OverallQoS(R_{qos})$ is the overall QoS score of all criteria of a user request. The weight W $\in$ [0,1] which represents the weight of each criterion and it is like a coefficient. Using SAW method, a score is assigned to each candidate Web service through multiplication of their QoS values with a user defined weight value. The value of weight shows the priority with reference to satisfaction of a given constraint provided by the user in the request. Weight is defined by the user which specifies the priority of the each QoS Parameter in the form of constraint. It should be between 0 and 1. This method selects the Web service which provides the good score for each task. The overall QoS score for candidate service can be expressed by value, which utilizes the values derived through scaling and weighting for the user request R for each criterion. The overall QoS score for candidate service can be represented as follows.

$$overallQoS(CS) = \sum_{\forall(q,v,w)\in\mathcal{R}_qox}(CS_p/CS_n)) * W \tag{2.9}$$

The equation (2.9) expresses the formula to calculate the overall QoS score by taking weighted sum of QoS values of selected candidate services to assign rank them for a composition.

Chapter 3

Related work

In this chapter, the existing work related to Web services research and application for Healthcare sector is presented. Section 3.1 discusses SOAP-based services discovery, selection and composition; Section 3.2 presents SOAP-based services discovery, selection and composition; Section 3.3 presents Cloud services discovery, selection and composition 3.4 summarizes the contributions made by various researchers.

3.1 SOAP-based Web services discovery, selection and composition

This section is divided into two subsections: first subsection describes the existing approaches related with my work, while second subsection presents the application of Web services-based approaches in Healthcare sector.

3.1.1 Approaches for Web services discovery, selection and composition

Among the related approaches, I have considered the work proposed by (Yu and Bouguettaya),(Zunino and Campo), (Mukhopadhyay and Chougule),(Ngan and Kanagasabai),(Dustdar and Schreiner),(Rao and Su),(Alamri, Eid, and El Saddik) for a comprehensive study on approaches for service discovery, selection and composition, specially semantics and QoS-based approaches.

An integrated approach proposed by (Yang and Papazoglou) towards service composition life cycle, which covers the service discovery, composition and selection of composed services without considering Semantic Web and QoS aspects.

Service discovery and composition issues of Web services in the proposed extended SOA (xSOA) architecture have been discussed by (Papazoglou and Van Den Heuvel). They have presented the concept of Semantic Web services to automate service discovery and composition process.

(i.) **Workflow-based Approaches**

The interleaving Web service discovery and composition approach was addressed in (Lassila and Dixit), by considering simple workflows, where Web services have one input and one output parameter. In this work, the Web service composition plan is restricted to a sequence of limited Web services corresponding to a linear workflow. In my framework, I propose a composition plan of services with multiple inputs and outputs, and also consider the other phases of the life-cycle of the service composition process, such as discovery and selection.

An algorithm for automatic composition of services was presented in (Zhang, Arpinar, and Aleman-Meza). The service composition is considered as a directed graph, where nodes are linked by the semantic matching compatibility (Exact, Subsume, PlugIn, Disjoint) between input and output parameters. Based on this graph, the shortest sequence of Web services from the initial requirements to the goal can be determined. This approach computes the best composition according to the semantic similarity of output and input parameters of Web services, but it does not consider any non-functional properties of the service composition.

An approach for Semantic Web Service composition has been presented by (Sirin, Hendler, and Parsia), which supports requesters to select Web services during each step in the composition process, and to create workflow specifications to link them. The discovery process consists of subsumption-based matching to filter services that can provide a concept as input for the current service at each step. This approach is semi-automated and totally dependent on human interaction. Therefore, it may be infeasible for large service repositories. Moreover, this approach does not consider QoS constraints.

The authors (Yu, Zhang, and Lin) proposed heuristic algorithms to find a near-to-optimal solution. The authors propose two models for the QoS-based service composition : 1) a combinatorial model and 2) a graph model. A heuristic algorithm is developed for each model. Despite the significant improvement of these algorithms, both algorithms do not scale with respect to an increasing number of Web services and remain out of the real-time requirements.

(ii.) **Genetic Algorithm-based Approaches**

Particle Swarm Optimization (PSO) and Genetic Algorithm (GA) are the representatives of Evolutionary Algorithm (EA). The PSO is a swarm intelligence technique first proposed in (Eberhart and Kennedy). By applying PSO to QoS-aware Web service composition, Web services would be viewed as particles in a N-dimension search space, which determined by the number of QoS indicators and constraints (Huang et al.). The author apply an improved PSO named as IDPSO to QoS-aware WSC by altering step size factor under different situation. The step size would be larger, when particles are far from optimal target; on the contrary, the step size would be smaller when particles approximate optimal target.

Genetic Algorithm is another popular technique that has been widely used in QoS-aware Web services composition (Weise et al.),(Ma, Wang, and Zhang),(Canfora et al.),(Fanjiang et al.),(Modi and Garg). The idea of GA inspired from the biological evolution, where duplication, crossover, mutation and selection are key functions. In GA-based QoS-aware Web services composition, solutions are encoded as chromosomes, where each code represents a service involved. In each iteration, the quality of solutions is evaluated and the top chromosomes chosen to perform crossover and mutation until the fitness requirement is reached. GA-based approach provides optimized solution with assurance, but it is a time consuming technique, so it is not appropriate to apply the approach for dynamic composition process. (Tang and Ai) has put forward a hybrid GA for seeking services with global optimal QoS value and least violation of constraints. A fitness function, punishment function and local optimizer are used to realize the optimization of population.

(iii.) **AI planning-based Approaches**

AI planning techniques have been proposed as a way to automate Web services composition (Carman, Serafini, and Traverso),(McDermott),(Sirin, Hendler, and Parsia),(Fadel),([McIlraith and Son),(McIlraith and Son).(McDermott) extended PDDL (Planning Domain Definition Language) (McDermott et al.) to deal with Web service composition. It was shown that the Optop (Opt-based total-order) planner (an extension to Unpop) could handle simple Web service composition. Even though McDermott proved that the estimated-regression planner is suitable for Web service composition, issues of dealing with non-determinism, partial observability and generation of conditional and iterative behaviors were not addressed.

SHOP2 (Sirin et al.) is a domain-independent planning system based on ordered task composition, which can greatly reduce the complexity of the planning process by avoiding some task-interaction issues. One of the shortcomings of SHOP2 is the inability to deal with non-deterministic actions. To resolve this problem, ND-SHOP2 planner (Kuter and Nau), a non-deterministic version of SHOP2, was developed. The main disadvantage of these approaches is that certain decomposition rules need to be encoded in advance with the help of process ontology. In order for decomposition rules, good knowledge of the domain is required.

([McIlraith and Son) and (McIlraith and Son) extended and adapted Golog for automatic Web service composition. They use formal methods to define, characterize and compute the preconditions and conditional effects for complex action. Service composition is done via the use of general templates that are modified based on user preferences. Later, they applied a modified version of ConGolog (Concurrent Golog) to the Web service composition problem (McIlraith and Son). Extensions were added to the ConGolog interpreter, so that it has the ability to implement sensing actions (for information gathering) as external function calls.

(Ponnekanti and Fox) presented SWORD for Web service composition, which uses a rule-based expert system considering the Rete algorithm (Forgy). SWORD faces several limitations, such as it does not support automatic service discovery.

Automated Web service composition was performed through planning as model checking, by modifying the Model-Based Planning (MBP) system (Pistore et al.). MBP receives as input Web services, represented as abstract processes in BPEL4WS, and a given goal process. It generates a description of the required composite service in BPEL4WS. This approach copes with issues, such as partial observability. Semantic information was not considered during composition.

A declarative approach (Gomadam et al.) was presented to compose Web services, using planning dynamically. This approach overcomes the issue of syntactic heterogeneities at the data and the functional level, through mediation based matching, describing semantic information in SAWSDL, and uses an extension of the GraphPlan to achieve a solution as a BPEL.

(Vuković, Kotsovinos, and Robinson) presented a framework that employs AI planning to rapidly assemble applications on-demand from individual services, based on context and user goals. The composition process utilizes the available services and the context, to create a composite service that satisfies the user's task intention.

(Hatzi et al.) proposed an integrated approach for automated semantic web service discovery and composition using AI planning techniques. In this approach, OWL-S Web service descriptions are transformed into a planning problem using PDDL. The implementation of approach is demonstrated by the development of the PORSCE II and VLEPPO systems.

(iv.) **Semantic Web-based Approaches**

(Da Silva, Pires, and Van Sinderen) developed DynamiCoS (Dynamic Composition of Services) framework to address all the steps and stakeholders of the dynamic service composition life-cycle. DynamiCoS supports end-users to perform automatic discovery, selection and composition using Semantic Web. They have developed prototype for healthcare scenario, which is similar to my prototype model. In this work, authors have not considered non-functional characteristics and keep that as a future scope. I have incorporated QoS parameters into service discovery, selection and composition process to enhance the user satisfaction.

(Rodriguez Mier et al.) presented a theoretical analysis of graph-based service composition in terms of its dependency with service discovery. Driven by this analysis a composition framework is defined by means of integration with fine-grained I/O service discovery that enables the generation of a graph-based composition, which contains the set of services that are semantically relevant for an input-output request. The proposed framework also includes an optimal composition search algorithm to extract the best composition from the graph minimizing the length and the number of services, and different graph optimizations to improve the scalability of the system.

(Küster et al.) proposed an approach to integrate service composition into service discovery and matchmaking to match service requests. Authors have discussed general issues involved in describing and matching services and present an algorithm implementing the work.

(Ngan and Kanagasabai) proposed a generic framework for Semantic Web Services (SWS) discovery, where discovery and selection are defined as the key tasks of the framework. They have described the benchmarks available to evaluate service discovery system; among these I have used WSC09 ([Kona et al.) datasets. Authors have focused on real life application of SWS discovery systems as an important problem with support of QoS feature. I have worked in the same direction to provide the solution.

(Bellur, Gupta, and Vadodaria) proposed various semantic matchmaking algorithms (i.e. Greedy, Bipartite matching & DL algorithm) based on functional and non-functional requirements of Web service. They have focused on comparative study rather than experimental evaluation of the algorithms.

The work of ([Zeng et al.), (Zeng et al.) focuses on dynamic and quality-driven selection of services by adopting global planning to find the best service components for the composition. They used (mixed) linear programming techniques to find the optimal selection of services. Similar to this approach, ([Ardagna and Pernici), (Ardagna and Pernici) extends the linear programming model to include local constraints. Linear programming methods are very effective when the size of the problem is small. Nevertheless, these methods suffer from weak scalability due to the exponential time complexity of the applied search algorithms.

(Laliwala et al.) proposed an approach, which Combines workflow, semantic and rules technology to achieve event-driven dynamic composition. This approach uses backward-chain methods for verification of precondition, input and output of services and forward-chain method with ECA rules for generation of composition schema as per BPEL standards.

METEOR-S (Verma et al.) is a Semantic Web-based framework for service composition, which offers mechanisms for semantic description of services, service discovery, and service composition. However, it is a static level composition using a template-based approach of processes. Opposite to this, I have focused on dynamic composition approach.

(Fujii and Suda) presented CoSMoS model for semantic description of services at different levels, i.e., at the data, semantic and logic for dynamic composition. In this approach, focus is given on composition process without considering performance metrics, while I have considered parameters like throughput and response time to measure the performance.

(Kona et al.) proposed an approach to perform discovery and composition of Web services semantically. A multi-step narrowing algorithm is used to perform the composition. Since Prolog with Constraint Logic programming are used to perform the discovery and composition processes, services are pre-processed from USDL (Sheng et al.) and transformed to Prolog terms. Pre-processing process is time consuming, especially for the dynamic service composition. They have not considered non-functional characteristic in the approach.

(Talantikite, Aissani, and Boudjlida) presented a model for automatic Web services discovery and its composition. In order to understandable descriptions, Semantic Annotation is used for Web service discovery and composition. The proposed approach uses from an inter-connected network of semantic Web services describing in OWL-S using the similarity measure between concepts like pellet before any submitted request. Their proposed approach gives several composition types: serial,dependent parallel and independent parallel. The Semantic Network is explored in backward chaining and depth-first in a single pass. At the end, several composition plans are obtained that satisfy the request and only one optimal composition plan using QoS is returned to the requester.

(Paikari et al.) proposed an automatic framework for composition of Semantic Web services in P2P network. The framework is modeled by Multi-agent System Engineering methodology, which is a famous agent oriented methodology and a top-down approach. This approach is consist four agents: UI Provider, Service Finder, Service Provider and Composer. The composition process is performed through several steps. At each step, composer sends its request for proper next Web service to a service finder.

An approach for QoS based Dynamic Web Service Composition have been proposed by (de Oliveira Jr and de Oliveira), which consists of three parts: Semantics, Syntax and Implementation. Semantics part contains Domain Ontology, Composer, Execution Engine and Service Repository modules. Authors have used a Greedy search approach, which is based on Greedy Algorithm to sort the list of known compositions in descending order through comparator function as a heuristic. Greedy search algorithm is a graph search algorithm, where a service is selected to be expanded based on a heuristic function. My proposed work on SOAP-based Web services is an extension of this work by integrating matchmaking and selection tasks with the service composition problem.

(v.) **Other Approaches**

(Mallayya, Ramachandran, and Viswanathan) proposed a QoS-based automatic web service composition framework by considering multiple QoS requirements as well as the user preferences. They also proposed a user preference based ranking algorithm, where user can specify his preferences over QoS parameters. This proposed approach composes Web services dynamically and generates the composition plan automatically. Although, authors have proposed automatic composition approach they have not mentioned the use of Semantic Web or AI technique in the proposed work.

(Gholam Hassan Tabatabaei et al.) proposed SCAIMO framework to fulfill security requirements of both service requesters and providers using secure task matchmaker. A prototype called SCAIMO-composer is developed for the validation of proposed work.

(Al-Masri and Mahmoud) proposed a Web service relevancy ranking function(WsRF) based on QoS parameters to find the best available Web services during discovery process based on a set of client QoS preferences. This work is focuses only on syntactic discovery process.

(Michlmayr et al.) proposed the QoS-aware Vienna Runtime Environment for Service-oriented Computing (VRESCO) for dynamic binding, invocation and mediation. Authors describe non-functional attributes in their service meta-data model. These QoS attributes can be specified manually using a management service or measured automatically and integrated into VRESCO environment.

(Vu, Hauswirth, and Aberer) presented a QoS-based Web service selection and ranking algorithm with trust and reputation management support. Authors have given a formal description and validation of the approach with experiments to demonstrate the quality of results under different cheating behaviors. This work mainly concentrates on the QoS-based Web service selection task.

3.1.2 SOAP Web services-based prototypes for Healthcare sector

(Hristoskova et al.) presents a Dynamic Composer for medical services. Based on a semantic description of the medical support services, this Composer enables a service to be executed by creating a composition of medical services. For this purpose, a system is implemented composing the medical services automatically through composition algorithms. The composition is achieved using various algorithms satisfying certain quality of service

(QoS) constraints and requirements. In addition to the automatic composition,the paper also proposes a recovery mechanism in case of unavailable services. When executing the composition of medical services, unavailable services are dynamically replaced by equivalent services or a new composition achieving the same result.

Process oriented semantic healthcare service composition framework using HL7 ontology and Semantic Web services has been proposed by ([Wang et al.). This approach performs service discovery and composition at design time in a semi-automatic manner which uses semantic profiles to specify the semantic descriptions of process operations using domain specific Ontology, such as HL7. This approach focuses on process oriented composition using WSBPEL which makes it less flexible and inappropriate for the dynamic environment of healthcare, whereas I have developed approach for dynamic and automated service discovery and composition.

(Lee et al.) proposed Semantic Medical Services (SMS) architecture for service discovery and composition using three types of compositional knowledge. Syntactic, semantic and pragmatic knowledge play a major role in discovery, selection and composition of Web services. They have implemented the framework in a cardiovascular domain. In this framework, authors have not considered user service request as well as performance evaluation.

COCOON is a web services based project aimed at reducing medical errors. This project focuses on resolving the problem of integration in healthcare domain through discussion of the problem of integrating components from service discovery to service composition. It is a WSMO compliant project and uses WSMO compliant service discovery engine for resolving the service discovery issue. In COCOON, the most appropriate services are discovered to be used by the specialist hence providing better healthcare services (Della Valle and Cerizza).

Artemis (Dogac et al.) is another project based on Semantic Web services for the semantic discovery and composition of services. It uses OWL-S as the approach for implementing semantic web services and uses HL7 as a standard for communication. Artemis uses OWL mapping tool (OWLmt) for the communication between sender and receiver providing semantic interoperability. OWLmt works as a mediator between sender and receiver by comparing sender ontology instances and receiver ontology instances with each other for making possible the communication (Della Valle et al.). The primary focus of Artemis project is on data interoperability aspect by resolving heterogeneities between HL7 standards V2 and V3.

3.2 RESTful Web services discovery, selection and composition

This section is divided two subsections. First subsection describes the existing approaches related with my work while second subsection presents the application of Web services-based approaches in healthcare sector.

3.2.1 Approaches for Web service discovery, selection and composition

The characteristics of RESTful Web services are fundamentally different than SOAP-based Web services, so approaches developed for SOAP-based Web service discovery, selection and composition are not suitable. Although the research community has put significant effort on automating SOAP-based Web services, automated RESTful Web service composition problem is less explored (Zhao and Doshi).

An approach presented by (Saquicela, Vilches-Blázquez, and Corcho) to automate the semantic description of RESTful services using a cross-domain ontology, DBpedia and external resources such as synonyms and suggestion services to annotate the parameters of the RESTful services. I adopt similar concept to annotate RESTful services. Another approach, SWEET (Maleshkova, Pedrinaci, and Domingue) facilitates semantic descriptions of Web APIs for developing mashups composed of Linked Open Data and Web APIs.

(John and Rajasree) proposed RESTDoc - that uses a microformat to describe, discover and composeRESTful services by adding semantics. They have defined two kind of discovery :(i) discovery as you browse for HTML link and (ii) Discovery as a service or automated discovery for linking related services, which enables automatic discovery and composition. The solution proposed by authors focuses on a microformat-like syntax to annotate the RESTful services and on an adapter for RDF conversion, whereas in my approach service discovery is achieved from a specified resource by generating the list of associated resources in the form of base URIs and subsequent graph construction mechanism is used to compose the resources.

RESTdesc (Verborgh et al.) represents the semantics of Web services using N3 format, and integrates existing standards and conventions, such as Link headers, HTTP OPTIONS, and URI templates for discovery and interaction. This approach is based on top of RESTful principles and consists of a semantic mark-up model. In the proposed approach, authors have concentrate on description and discovery of services only.

(Pautasso) proposed an extension to BPEL for a composition of REST and SOAP Web services. REST services are wrapped in WSDL descriptions to generate a BPEL composition. My proposed approach highlights a native composition of RESTful services, rather than using technologies of SOAP Web services.

(Bennara, Mrissa, and Amghar) proposed an approach for service composition based on the REST and Linked Data. The proposed solution depends on the description of web resources to support a unified discovery process and composition, where the interactions among resources can be tracked themselves, and publish these traces, through the progressive exploration of resource descriptors. In my approach, I have followed the same principle for discovery and composition with incorporation of additional information on resources such as QoS value for the selection of matched resources.

iServe (Pedrinaci et al.) provides platform for publishing services for discovery and use by exposing service descriptions as Linked Data expressed as a simple vocabulary for services, such as SOAP-based services and RESTful services. In my work, Linked Data principles are used for discovery, selection and composition tasks.

(Zhao and Doshi) explored the RESTful service composition problem using a situation calculus based State Transition System (STS) as a formal model to automate the composition process of RESTful Web services. Authors have identified the challenges of resource description language and data heterogeneity. I have considered LOD based RDF model to describe the RESTful Web services and dereferencing mechanism to compose them.

Algorithms for automatic Web API discovery and composition are proposed by Yong-

Ju Lee et al. (Lee). The discovery algorithm can generate optimal plans by applying strategies that rapidly prune APIs that are unmatched to the query, while the composition algorithm is based on a graph-based approach, where composition candidates are gradually generated by forward and backward searching over the graph. This work considers discovery and composition of Web APIs, such as SOAP and REST using ontology based approach while my proposed approaches cover discovery, selection and composition of RESTful service using Linked Data principles to get the benefit of REST.

Data-Fu (Stadtmüller et al.) enables the development of data driven applications that facilitate the RESTful manipulation of read/write Linked Data resources. For interactions between web resources, this work introduces a lightweight declarative rule language called Data-Fu with state transition systems as formal grounding.

Karma (Knoblock et al.), a semi-automatic approach presented for mapping structured sources to ontologies in order to build semantic descriptions as source models. It also provide a graphical user interface for a user to interactively refine the models, which can be used to convert data into RDF with respect to a given ontology or to define a SPARQL end point to work with an ontology. Demonstration of Karma tool allows users to extract, link, and integrate geospatial data from various sources.

REST-based method is used by Bohara et al. (Bohara, Mishra, and Chaudhary) to solve certain limitations, such as tightly coupled invocation, performance in terms of response data, non-uniform interface and no hyperlink support. In this work, Web Processing Service interface specification provides an approach to publish and execute geo-processes on the Web. Geo-spatial analysis plays critical role to generate alerts and recommendations. Authors have provided android based user interface with local language support but doesnt provide performance measurement detail of the proposed algorithm.

To minimize the manual efforts and to get the results faster and automated, Alice et al. (Alice, Abirami, and Askarunisa) and (Zhang et al.) proposed the use of Semantic Web Technology like, OWL, RDF and SPARQL to retrieve the information from the heterogeneous documents efficiently. These works propose a model for storing the content of a DOC/HTML document in the RDF format and thereby enabling the information retrieval using SPARQL. Efforts have been done for information retrieval rather than discovery and composition, which are main objectives of my work.

Page et al. (Page, De Roure, and Martinez) presented an approach about how web service gives semantic results using RDF and linked open data. In this work, author has discussed about domain driven approach without representing framework and algorithm detail.

Mouhoub et al. (Mouhoub, Grigori, and Manouvrier) presented a SPARQL-driven approach for searching linked data and relevant services. An approach for service discovery and composition has been proposed for LOD-based SWS services. Author has not presented implementation and experimental detail of the work.

Hypermedia-Driven RESTful service composition (Alarcon, Wilde, and Bellido) performs dynamically when a client interacts with a server. This approach utilizes a hypermedia-centric REST service description called the Resource Linking Language (ReLL) and Petri Nets as a mechanism for describing the machine-client navigation.

OmniVoke (Li et al.) presented the framework for automated invocation of Web APIs. This work provides a framework that supports the invocation of most Web APIs available over the Web. This framework focuses on semantic description of Web APIs using the extended service model. The presented work focuses on semantic annotation and invocation part, whereas I focus on discovery and composition of Web services.

3.2.2 RESTful Web services-based prototypes for Healthcare sector

Ji et al. (Ji) has shown that how to extract patient-generated or reported social health data, open government health data, and clinical research data from heterogeneous data sources, and then link and combine it to develop an integrated semantic network model. In this work, framework is presented but they haven't given any detail about proposed approach.

Sheth et al. (Sheth, Anantharam, and Thirunarayan) considered semantic perception algorithms to transform raw data to human intelligible abstractions using medical domain knowledge. A framework called Knowledge-enabled Health(kHealth) is proposed to empower an individual in their health and fitness needs. The key features of kHealth are based on use of (a) relevant personal, public and population level health signals, (b) Semantic Web technologies that include application of relevant medical knowledge in the forms of ontologies & reasoning and (c) intelligence at the edge by performing all computation on the users mobile device.

(Parra et al.) proposed a RESTful discovery and eventing specification based approach, which is implemented in the context of an assisted living healthcare scenario for connecting heterogeneous sensor and actuator devices in order to obtain real-time health data from the inhabitants for better health monitoring.

Andry et al. (Andry, Wan, and Nicholson) presented the REST-based architecture for creating a REST API integrated with a mobile application (iPhone/iPad) that offer physicians, access to their patient's health records via a community, regional or state Health Information Exchange (HIE). They describe the architecture of the system which address security and privacy concerns, the REST API operations and HL7 subset data format used for lab results and observations.

3.3 Cloud services discovery, service selection and service composition

This section presents previous studies related to cloud service tasks, such as matchmaking, selection, and composition. To support the present research, we consider earlier works related to cloud services, the proposed approaches to cloud service tasks, and the usage of cloud services in the healthcare sector.

3.3.1 Approaches for Web service discovery, selection and composition

Joshi et al. (Joshi, Yesha, and Finin) proposed a life cycle model of cloud services with five phases: requirements, discovery, negotiation, composition, and consumption. The authors presented an integrated ontology to automate the cloud service lifecycle and provided a cloud storage service case study to find and achieve storage on the cloud by presenting the service attributes and policies. Our proposed approach was inspired by this previous work and others related to the Web service composition life cycle.

Ngan et al. ([Ngan and Kanagasabai) proposed the system supports dynamic semantic matching of Cloud services described with complex constraints and brokering policies currently in use for matching the user need at runtime, as well as the service usage cost. The approaches are based on formal description of service functionality using the SAWSDL, OWL-S and WSMO description languages.

Han et al. (Han and Sim) presented Cloud Service Discovery System (CSDS) for users to perform discovery over the Internet, that consults an ontology when retrieving information about Cloud services. This work focuses on discovery process whereas, I have developed discovery mechanism by incorporating non-functional parameters.

(Sun et al.) carried out a survey of approaches to cloud service selection by considering the seven perspectives related to context, selection and criteria weight. After analysing the reviewed approaches based on these seven perspectives, research issues related to contemporary cloud service selection were identified.

(Wang et al.) proposed QoS-aware service selection approach. Authors have focused on importance of QoS parameters as a part of SLA to select efficient services and assign ranking for service selection. I have used the QoS model provided by (Wang et al.) to propose improved approach for service selection.

(Garg, Versteeg, and Buyya) proposed the Cloud service ranking framework by evaluating Cloud services using Cloud Services Measurement Initiative Consortium (Siegel and Perdue), where Cloud services are ranked using the analytic hierarchy process technique (AHP), which consists of three main steps- problem decomposition, priority judgment and priority aggregation.

(Zhang et al.) proposed a system called CloudRecommender for the automatic selection of infrastructure cloud service configurations. The CloudRecommender supports a selection based on previous stored service information but does not allow a runtime selection using dynamic QoS information.

Some researchers have worked on the Cloud services selection problem with a specific service type, such as SaaS, IaaS, or PaaS. Godse and Mulik (Godse and Mulik) have proposed an AHP-based SaaS service selection method to score and rank services. An AHP hierarchy is prepared to define SaaS service attributes. The relative importance of service attributes is weighted by adding user preferences and domain experts? opinions.

Karim et al. (Karim, Ding, and Miri) presented the techniques for aggregation of multi-level QoS parameters of Cloud services, which supports the ranking and selection of IaaS and SaaS services based on user requirements. In this approach, an AHP hierarchy of a Cloud service weighting model is defined, where QoS criteria of both SaaS and IaaS are considered for the ranking. The proposed AHP hierarchy allows the generation of QoS weights, and the usage of the mapping rules to achieve a best service combination of IaaS and SaaS.

Menzel et al. (Menzel, Schönherr, and Tai) developed a CloudGenius for selection of best combination of a VM image and a cloud infrastructure service to support web server migration to the cloud for IaaS selection. The proposed approach is based on analytic hierarchy process (AHP). The prototype implementation presented is limited to criteria like popularity, price and performance and supports only Amazon-web services.

(Jrad et al.) presented a utility-based approach for Cloud services selection through functional and non-functional requirements, such as availability, response time and throughput. I have extended this approach and proposed new mechanism for Cloud service selection.

Jrad et al. (Jrad, Tao, and Streit) proposed architecture for a Cloud service broker for Intercloud environment, where broker aims to find the suitable cloud provider while satisfying the userśservice requirements in terms of functional and non-functional SLA parameters. In this work, authors have proposed matchmaking approach for cloud services. I have extended this approach and proposed improved work for Service discovery using QoS as SLA parameters.

From the perspective of cloud computing, (Sim) contributes to the field of cloud resource management by devising several novel approaches for facilitating cloud service discovery, service negotiation, and service composition. Their work included the development of agent-based search engine, negotiation technique and cooperative problem-solving technique to automate the cloud service composition.

An approach for workflow engine was proposed by (Höing) to integrate different Cloud Computing platforms and services, offering Orchestration as a Service (OaaS). However, the prototype implemented had some missing features, such as automatic service selection, mechanisms for failure/fault handling and adaptability at runtime, use of quality metadata, and SLA support.

Cavalcante et al. (Cavalcante et al.) presented Cloud Integrator for the service composition offered by various Cloud Computing platforms. In Cloud Integrator composition of services is represented using a semantic workflow. This approach focuses on the composition mechanism of the platform. The metadata-based service selection algorithm and its application is described in a case study.

A novel quality-based cloud service composition approach using QoS attributes proposed in (Ye, Bouguettaya, and Zhou). In this work, the tenure process of university department is considered to demonstrate the cloud service composition. This work focuses on syntactic aspects of composition rather than semantic composition.

Gutierrez-Garcia et al. (Gutierrez-Garcia and Sim) proposed an agent-based technique to composing services in multi-cloud environments. Agents are enabled with a semi-recursive contract net protocol and service capability tables to compose as per the user requirements. The single and multiple cloud schemes applicable to dynamic service composition are not provided.

A generic methodology for the representation of Cloud services was proposed by (García and Blanquer), which uses the WS- Agreement specification for capturing and manipulation arbitrary services using SLA fragments. A SLA composition algorithm composes SLA fragments on the fly in response to user request. This work also focuses on syntactic aspects of composition rather than semantic composition.

(Ye et al.) proposed a Cloud service composition framework, that selects the optimal composition based on an end userś long-term Quality of Service (QoS) requirements. The proposed framework uses a new multivariate QoS analysis to predict the long-term QoS provisions from service providers´historical QoS data and short-term advertisements represented using Time Series. To select the optimal service composition, the proposed framework uses QoS time series inter-correlations and performs a novel time series group similarity approach on the predicted QoS values.

Ye et al. (Ye, Bouguettaya, and Zhou) presented a novel quality-based cloud service composition approach. The research focuses on the selection of composition plans based solely on non-functional (QoS) attributes. It proposes a cloud service composition approach to aid end users selecting and composing SaaS providers and IaaS providers in the cloud environment.

Guidi et al. (Guidi, Giallorenzo, and Gabbrielli) presented an APIasaService (API-aaS) layer as a tool to ease the development and deployment of applications based on the API composition. However, the authors focused only on the development and reuse of the API composition and did not carry out experimental studies.

3.3.2 Cloud services-based prototypes for Healthcare sector

Kuo and Alex (Kuo) discussed the demand for a cloud computing platform in the healthcare domain, as well as issues regarding its management, technology, security, and legal aspects, along with case studies. They proposed the HC2SP model for the application of such platform in the field of healthcare.

Parekh et al. (Parekh and Saleena) proposed a framework, which is based on main components of healthcare system: patients, doctors, symptoms and diseases. It enables patients to get required health services, such as to book an appointment with the doctor and to get disease information through data mining clustering techniques and mobile application. This approach has focused on statistical analysis of historical data whereas, I have focused on remote treatment facility based on diagnosis resulted from patientś symptoms and samples using Cloud services for the beneficial to end users.

3.4 Summary

The literature review outlines some remarkable observations, such as Web services composition is still interesting research challenge and it is closely related to the service discovery and selection processes to perform the composition efficiently and accurately. Very few efforts have been made to integrate the service discovery, selection and composition for SOAP-based Web services, i.e. [(Yang and Papazoglou),(Da Silva, Pires, and Van Sinderen)] in these approaches, they have not considered non-function requirements which add user satisfaction and quality guarantee to the solution. Heterogeneous Web services provide features, such as dynamic in nature, interoperability, support for Web and heterogeneous environment makes it more suitable for healthcare domain.

Through literature study, I have observed that service discovery, selection and composition are interrelated concepts although little attention has been given to integrate them to enhance the performance and efficiency of the solution using RESTful and Cloud services. Moreover, the application of RESTful services in healthcare sector is still very few.

All the aforementioned facts motivates to develop frameworks and solutions for Heterogeneous Web services by integrating service discovery, selection and composition processes. Moreover to demonstrate the applicability of the work, there is a need to develop prototypes by considering available datasets and developing repository of real Web services.

Chapter 4

SOAP-based Web services Discovery, Selection and Composition

In this chapter, the proposed framework and approaches for service discovery, selection and composition and experimental work of SOAP-based Web services have been presented [1],[2],[3]. Section 4.1 discusses proposed framework for SOAP-based services discovery, selection and composition; Section 4.2 presents SOAP-based Web Service Composition Problem; Section 4.3 presents Web Service Matchmaking Algorithm; Section 4.4 presents Web Service Selection; Section 4.5 presents Experimental Work and Results; Section 4.6 presents Prototype for Healthcare Information System; Section 4 concludes the chapter.

[1]Part of this chapter has been published as Kirit Modi, Sanjay Garg, *An Approach for Ontology based Web Services Composition*, Intelligent Systems and Control (ISCO2011), 2011 5th International Conference on. 2011.

[2]Part of this chapter has been published as Kirit J. Modi,Sanjay Garg, *Dynamic Web Services Composition using Optimization Approach*, International Journal of Computer Science & Communication 6 (2015), 285-293.

[3]Part of this chapter has been published as Sanjay Garg, Kirit Modi Sanjay Chaudhary *A QoS-aware Approach for Runtime Discovery, Selection and Composition of Semantic Web Services* ,International Journal of Web Information Systems(IJWIS), Vol. 12 Iss: 2, pp.177 - 200

4.1 Proposed framework for service discovery, selection and composition

The proposed framework represents the relationship among discovery, selection and composition tasks of Web services as shown in figure 4.1. The operation of the framework can

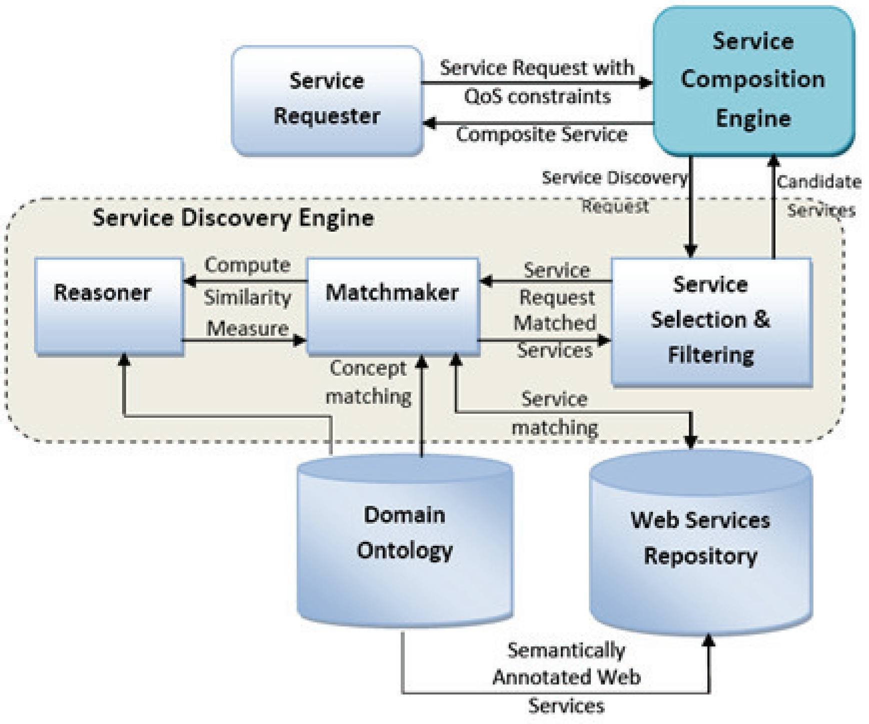


Figure 4.1: Proposed framework for Web services discovery, selection and composition

be described as follows:

- A composition process begins, when a Service Requester requests some functionality from service. Service Requester provides input parameters of service, required output and Quality of Service constraints as part of composition request.
- Composition Engine starts to discover a set of services, that provide required output and fulfill QoS constraints imposed by Requester from Service Repository with the help of Matchmaker and Service Selection & Filtering module.
- Matchmaker module performs matchmaking operation between Userś Query and

Services provided by Repository. Reasoner is used with Matchmaker for computing similarity measure of the concepts. Matchmaker provides matched services to the Service Selection & Filtering module for selection, sorting and filtering of candidate services for composition. QoS parameters, i.e. Throughput and Response Time are used for filtering purpose. The matchmaking process is performed using domain ontology.

- Composition Engine is based on Composition algorithm. It generates composition plans using candidate services and provides best composition solution as an executable composite service.

The proposed framework consists of following components.

- **Service Requester:** Service Requester may be a person or a computer requests some functionality as required output by providing inputs along with QoS constraints.

- **Service Composition Engine :** It is responsible to compose candidate services provided by the Matchmaker and Service Selection & Filtering module. Composition engine interacts with Requester as well as Execution Engine. The output of this component can be an integration plan, a workflow, or a reference to a composite web service.

- **Service Selection & Filtering Module :** Service Selection & Filtering module is responsible to select services which can be used as service compositions. After selection, services are filtered based on requesterś QoS constraints. Overall quality constraint provided by requester would be compared for overall quality score of composite services for filtering. The selection process correlates various types of service descriptions (such as semantic, syntactic, QoS) at runtime to reach the best composition of Web services that suits the userś need.

- **Reasoner :** A Reasoner uses the Domain Ontology database to compute the similarity measure of the concepts for semantic matchmaking process. The Reasoner performs semantic match using concepts of input parameters and output parameters of Web service during the matchmaking process.

- **Semantic Matchmaker :** The semantic matchmaker is used to perform matchmaking between user request with the set of services available in Web service Repository. If the service is matched with the requested service then it passes the result to the service filtering module which contains list of matched services as per the user request .

- **Web Services Repository :** This repository contains the semantically annotated Web services which is connected with the matchmaking module so that the available Web services can be discovered from the repository. Qos information is associated with the Web services. Each Web service provides Response Time and Throughput as QoS parameters.

4.1.1 Comparison of SUPER framework with Our Approach

This section summarizes the comparison of SUPER framework with the proposed approach.

- The framework implemented in the SUPER (Semantics Utilized for Process management within and between EnteRprises) project (Hepp and Roman) is similar to Our Approach in terms of key components and their functionalities (Karastoyanova et al.).

- The major objective of SUPER was to bridge the gap between the business needs expressed by business people and the actual Information Technology (IT) infrastructures intended to support them, while also supporting in a more efficient way the reuse and automation of business processes. For this reason, it implements a semantic-based and context-aware framework platform that supports the management of business processes, through the use of Semantic Web servicesťechnologies.

- The final platform of SUPER architecture includes modules for the automated discovery, composition and execution of business process implementations. Furthermore, three use case scenarios were developed for the needs of the project, all based on the telecoms domain, covering the fields of fixed telephony, traffic routing and the management of mobile environments. Despite the different objective of SUPER project, it shares similarities with respect to Discovery and Composition modules in comparison with the proposed approach.

- SUPER project includes heuristic-based forward search AI planning approach, while I have used Greedy search approach for composition. The major drawback of AI-planning based approach is that they do not perform well in situations where the number of participating web services is large whereas Greedy search approach provides efficient solution for various levels of services within reasonable time.

- Information related to the non-functional properties (QoS) is not provided by the SUPER project whereas, I have used Response Time and Throughput as QoS parameters in the proposed work.

In order to develop the composition algorithm, basic terms related to web service composition are defined as follows.

Request (R_q): A composition request Rq consists of a set of provided inputs R_{qin}, a set of required outputs R_{qout} and a set of QoS constraints R_{qos} imposed by the requester.

Domain Ontology (Ont): A Domain Ontology Ont consists of a set of concepts used to describe the domain in which a group of services is inserted. These concepts are related to each other according to subsumption relation, i.e. a concept $c1 \in Ont$ subsumes another concept $c_2 \in Ont$ if c_1 is a superclass of c_2; c_2 subsumes c_1 if c_1 is a subclass of c_2; c_1 and c_2 subsumes each other if they are the same.

Web Service (S): A Web service S is defined as $S_i, , S_o$, where S_i = $S_1, S_{2,,Si}$ is a finite set of required input concept and S_o= $S_1, S_2, , S_j$ is a finite set of provided output concept.

Service Repository (SR): A service repository SR consists of a set of available services. Each service $S \in SR$ is composed of a required inputs S_i, a set of provided outputs S_o,and, a set of QoS criteria S_{qos} = $(q_1, v_1); (q_2, v_2), \ldots.., (q_k, v_k)$, where q_i (i = 1, 2,..., k) is a quality criterion, v_i is the value of the criterion q_i provided by the service, and k is the total number of criteria provided.

Service Composition (SC): A service composition SC is a Directed Acyclic Graph (DAG) with vertices C_v = $S | S \in SR$ and edges C_e = (u ,v)$| \forall u, v \in C_v \wedge \exists c_1 \in u_{out} \wedge \exists c_2 \in vin$

: c_2 subsumes c_1

Comparator function (C_{cmp})**:** A comparator function (C_{cmp}) sort the service composition (SC) and indicating whether SC1(r <0) or SC2 (r >0) should be expanded next.

Sorted list of composition (X): A sorted list of composition X contains set of compositions Sorted according to comparator function.

RequestersQ́oS Constraints (R_{qos})**:** a set of QoS constraints provided by Requester for desired services.R_{qos} = $(q_1, v_1); (q_2, v_2), ..., (q_k, v_k)$, where qi (i = 1, 2,...,k) is a quality criterion, vi is the value of the criterion qi provided by the requester, and k is the total number of criteria provided.

Semantic Similarity between Concepts: The semantic similarity between concepts refers to the degree that the two concepts, i.e. outR (a required concept) and outS (a service concept)can match, and it is a quantitative definition given on the basis of the four match types (Bellur and Kulkarni). The calculation formula is as follows.

- **Exact:** If outR and outS are exactly the same concepts;
- **Plug-in :** If outR is subsumed by outS, means outR is a subclass of outS;
- **Subsumes:** If outR is subsumes outS, means outR is a superclass of outS;
- **Fail:** when subsumption relationship between outR and outS is not available.

An algorithm for Web service composition shown in Algorithm 1 is inspired by the work proposed in (de Oliveira Jr and de Oliveira). The Algorithm 1 performs a service composition for a given request R_q. It starts by searching all services which can provide as output a concept which is semantically equivalent to the required outputs specified in $outR \in R_{qout}$. The matchmaking process is performed to calculate semantic similarity measure for a given request against available services. Matched results will be used for selection of candidate services (lines 2-8). A service will be a considered as candidate composition if it satisfies QoS constraints specified by user request R_{qos}. Quality parameters Q_{qos} are extracted from the services and compared with quality constraints imposed in the request to get overall value of QoS. Candidate compositions are filtered those no longer

Algorithm 1 Web Services Composition Algorithm

```
Input: $Rqout \rightarrow CompositionRequest, SR \rightarrow ServiceRepository$
Data:$X \rightarrow thesetofcandidatecompositionsOutput : SC \rightarrow thecomposition, or\Phi$
 1: for each $outR \in R_{qout}$ do
 2:     for each $S \in SR$ do
 3:         $S \leftarrow matchmaking(outR, outS)$;
 4:         $SL \leftarrow selection(S, Rqos)$
 5:         $SCv \leftarrow SL$;
 6:     end for
 7: end for
 8: for $EachS_{qos} \in SC$ do
 9:     if $OverallQos(S_{qos}) > OverallQos(R_{qos})$ then
10:         $X \leftarrow append(X, SC)$;
11:     end if
12: end for
13: while $X \neq \Phi$ do
14:     $X \leftarrow sortServices(descending, X, C_{cmp})$;
15:     $SC \leftarrow popLastComposition(X)$;
16:     if isSolution(SC) then
17:         return SC;
18:     end if
19:     for each $S \in SR$ do
20:         $S \leftarrow matchmaking(outR, outS)$;
21:         $SL \leftarrow selection(S, Rqos)$
22:         $SC_{vnew} \leftarrow SCvUSL$;
23:         $SC_{enew} \leftarrow SCeU\forall S2 \in cv(S, S2)$;
24:         if $OverallQos(S_{qos}) > OverallQos(R_{qos})$ then
25:             $X \leftarrow append(X, SC_{new})$;
26:         end if
27:     end for
28: end while
29: return $\Phi$;
```

meet the quality constraints (lines 9-13). The filtered candidate compositions are appended into X.

The candidate compositions are sorted in descending order using comparator function which compares two candidate compositions SC1 and SC2 and return a value (r) below zero if SC2 is more promising than SC1, above zero if SC1 is more promising than SC2 and zero if both are equally evaluated. The comparator function proposed in (Weise et al.) considers four composition properties: (i) known concepts-consists of input concepts provided by the requester and output concepts provided by the output of all services in SC; (ii) unknown concepts-consists of output concepts of the requester and input concepts of each service; (iii) eliminated concepts-consists of already provided unknown concepts; and (iv) the number of services in composition.

The comparator function considers overall QoS along with other four composition properties. Once the list is sorted using the comparator function, the composition is returned as a solution which has high overall QoS score (lines 14-19). Otherwise, the chosen composition is expanded to form new candidate compositions (lines 20-28). The algorithm runs until a candidate solution is found or all candidate solutions are expanded and rejected.

4.1.2 Algorithm Analysis

This section provides time complexity of the algorithm as follows. Time = O(p^3), Where, p is the number of services participating in the composition. The algorithm proposed (de Oliveira Jr and de Oliveira) have shown time and memory complexity O(p^m), where p is the maximum number of services participating in the composition (the number of services that can produce any concept required by the candidate composition) and m is the maximum number of services in a composition. Similar time complexity is achieved for this proposed approach.

In the best case, the time complexity of my approach is O(p^2), it defines the situation when composition solution will be found at first attempt. In the worst case and average case, the time complexity is O(p^3). It define the situation, when all possible candidate services participating in the composition required to be considered.

4.2 Web Service Matchmaking Algorithm

The matchmaking algorithm presented below is proposed by Choi et al. (Choi, Han, and Abraham) and used in (Bellur and Kulkarni). The same algorithm is used here as a matchmaking process as shown in Algorithm 2.

Algorithm 2 Web Services Matchmaking Algorithm

Input: outR (required concept), outS (published concept)

Output: Exact or Plug-in or Subsumes or Fail

```
1: if $outR = outS$ then
2:     return Exact;
3: else
4:     if outS subsumes outR then
5:         return Plugin;
6:     else
7:         if outR subsumes outS then
8:             return Subsumes;
9:         else
10:            return Fail;
11:        end if
12:    end if
13: end if
```

For the matchmaking Algorithm 2, the best case, worst case and average case time complexity is O(1). It shows constant value for the time complexity.

4.3 Web Service Selection Algorithm

Set of Discovered services (SD): Represents services retrieved through discovery process. Userscriteria (Cj): Defines a criteria specified by the user through request. Normalized value of $QoS(N_{qos})$: Provides normalized value of QoS parameters.

Algorithm 3 Web Services Selection Algorithm

Input: R_{qos}(User Request), SD (Set of Discovered services)

Output: SL (Sorted list of Selected services)

```
 1: Begin
 2: $S \rightarrow S_{qos}$;
 3: for each $Si \in SD$ do
 4:     if overall $S_{qos} > R_{qos}$ then
 5:     end if
 6: end for
 7: while $i \leq n$ do
 8:     for $j = 1 to 2$ do
 9:         if $qi(Si) < Cj$ then
10:             $filteroutSi$;
11:         end if
12:     end for
13: end while
14: for each $Si \in SD$ do
15:     calculate CSp & CSn;
16:     Qscore = $N_{qos}$ + w ;
17:     $SL < sort(Q_{score}, desc)$;
18: end for
19: return SL
20: End
```

In the Algorithm 3, line 2 to 5 represents the list of candidate services for requesters with their QoS values. Line 6 to 11 represents the filtering mechanism. During the filtering process, matching of the user request criterion (cj) with each service criterion (qij) is performed. The services, which are unable to satisfy the requesterś specified constraints will be filtered out. The aim of the service filtering process is to generate refine result to the composition module to increase the quality and performance of the composite result. Line 12 to 15 represents ranking mechanism and select the best services according to highest QoS score. The ranking process is performed as follows. First of all scaling method is applied to get normalize value of all QoS parameters for each service then after the weighted value (w) is calculated. The total QoS score will be calculated by combining the normalized value and weighted value. Once the total score is computed, sort the services in descending order according the total QoS score. The selected list of services will be generated as result of the service selection process.

4.4 Experimental Work and Results

This section describes the detail about the experimental work carried out and results, which are generated based on the experiments. All the experiments presented here have been performed on with Intel Core 2 Duo 2.0 GHz processor with 3.0 GB RAM memory, Eclipse Europa with JDK 1.6.0, protégé 4.3, Jena API.

In order to evaluate the proposed approach, I have used the Web Service Challenge (WSC 2009) dataset, which is a collection of five different datasets of semantically enabled Web services scaled from, 136 to 8016. These Two QoS parameters, such as response time and throughput are associated with these services to represent the quality aspects demanded by the user. In the following experimentation, the effect of increasing the size of Web services on the performance of proposed work is presented.

I have measured the execution time of algorithms by running them at least 5 times validation of results and get the average value of them. A sample query could be: response time¡1000ms, throughput $\geq$ 25. Then the query vector is set as: (1000, 25). A sample preference vector for this query could be: (1, 2).

4.4.1 Performance Parameters

Following performance parameters have been considered to measure the performance of the approach

(i) **Number of Services:** It defines total number of services available in the repository.

(ii) **Execution Time:** It represents the time required to perform the composition process.

4.4.2 Comparison of Proposed Approach with QoS-based Approach

A comparison of results of proposed approach with QoS-based approach (de Oliveira Jr and de Oliveira) is depicted in Figure 4.2. Results show that proposed approach performs better in comparison with existing QoS-based approach.

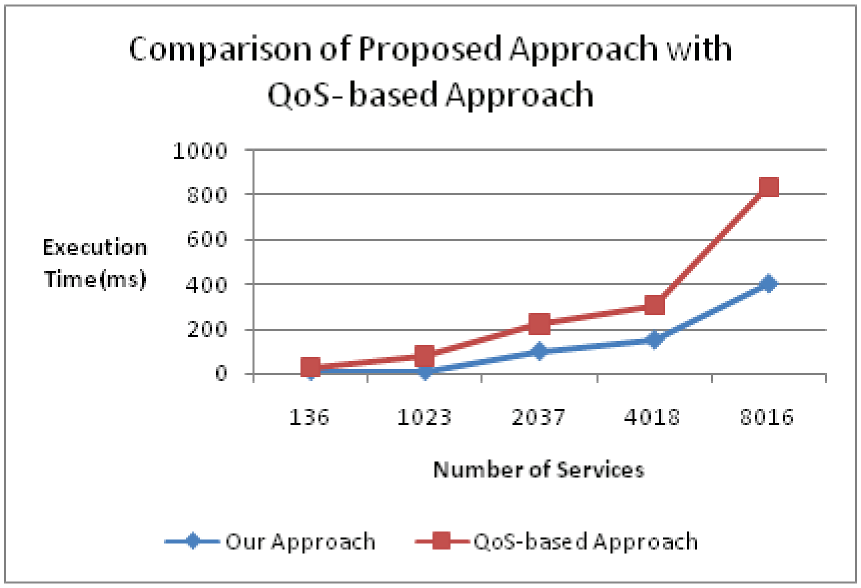


Figure 4.2: Comparison of Proposed Approach with QoS-based Approach

The results show that for different values of services of dataset the proposed approach performs well in comparison with the QoS-based approach, which has focused on composition process only while to make an approach more efficient integrated approach is used in the proposed work.

4.4.3 Comparison of Proposed Approach with Hybrid Approach

A comparison of proposed approach with Hybrid approach (Ma, Wang, and Zhang) is presented in Figure 4.3.

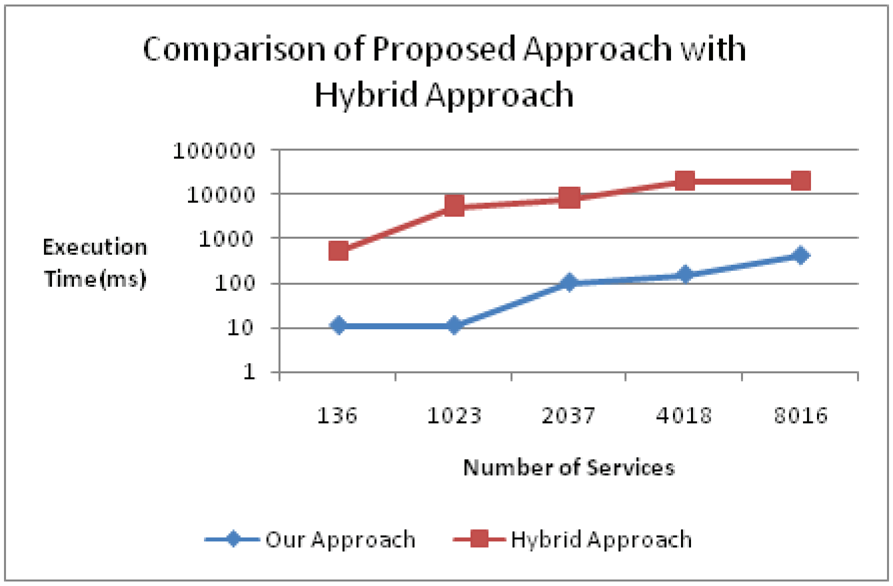


Figure 4.3: Comparison of Proposed Approach with Hybrid Approach

The Hybrid approach is an integration of Genetic programming technique with Greedy search method, which increases execution time of composition process while my approach is based on greedy search technique by integrating discovery, selection and composition process, which performs composition process in reasonable time and the generated solution is found to be near to optimal solution.

The following section discusses the prototype of Healthcare Information System (HIS). The purpose of this prototype is to validate the proposed framework. With this prototype, the applicability of the proposed work in the healthcare sector is presented. A comparison with existing prototype is also presented.

4.5 Prototype for Healthcare Information System

A Healthcare Information System (HIS) integrates the healthcareś business process, and information systems to deliver better healthcare services[1]. The Healthcare information is offered through electronic medium, i.e. EHR (Electronic Health Record) but different organizations uses different formats, which create problems of standardization and interoperability of the information. Some key challenges of EHR adoption are cost, ownership, standards and human factors. An ideal solution is required to make healthcare information interoperable and easily accessible by the end users. Another requirement is sharing of information for better and quick solution. By considering these challenges, I have developed a prototype for Healthcare Information System using the proposed approach as shown in Figure 4.4. The main components of the prototype framework are: discovery engine, selec-

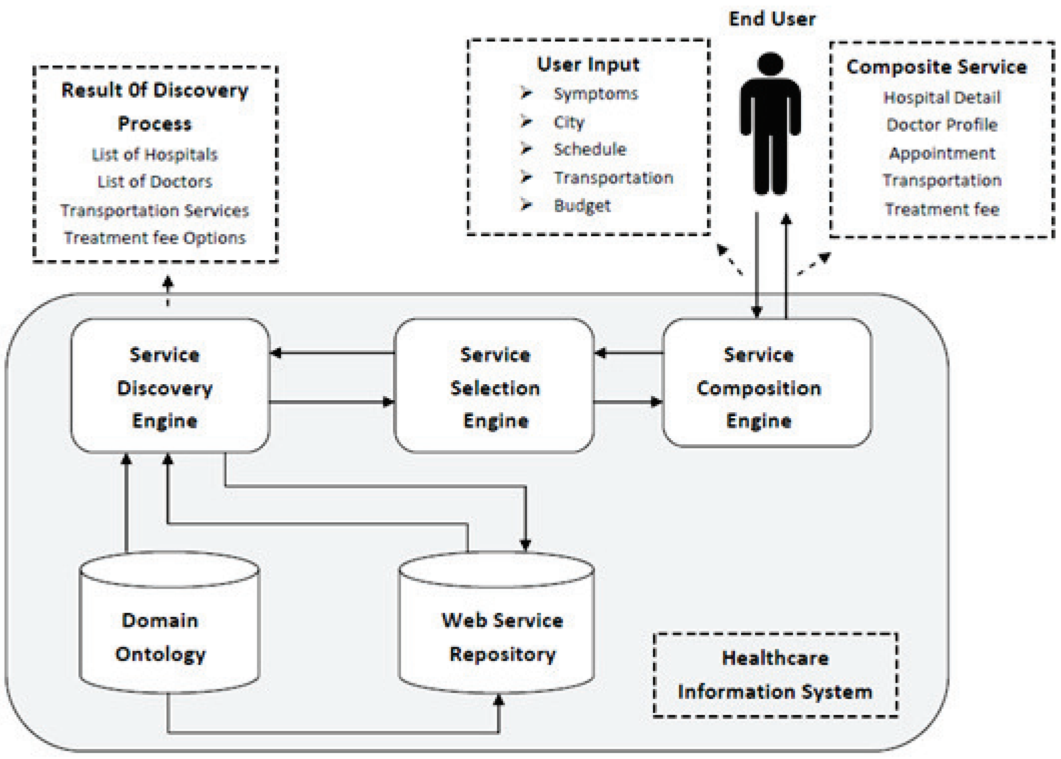


Figure 4.4: A prototype for Healthcare Information System

tion & filtering engine and composition engine. At first, usersŕequirements are translated into XML based format using request formulation module and forwarded the request to the discovery engine.

The Discovery engine will perform semantic matchmaking based on usersś request and semantically enabled services derived from the web service repository. Discovered services will be utilized for selection and filtering based on user specified QoS constrains. The resultant services are composed through composition engine and composite services will be delivered to the user as a result.

4.5.1 Web Services for the Healthcare Information System

In order to describe the dynamics of the prototype framework, a use case from the electronic-healthcare domain is presented here. I define several Web services as shown in Table 4.1 related to Healthcare sector. List of Web services as shown in Table 4.1 contains semantically enabled services whose input and output parameters are mapped with concepts of the domain ontology of dentistry domain of Figure 4.5

Table 4.1: List of Web services

Sr. No.	**Functionality of the Web service**	**Inputs**	**Outputs**
1.	getHospitalWithDoctor	Symptoms, Location	Hospital, Doctor
2.	getTransportation	Pickup-point, Landmark	Transport facility
3.	getAppoinment	Date, Time	Appointment
4.	getPackage	Consulting Fee	Payment
5.	getOffer	Offer inquiry	Offer detail

The Web services such as getHospitalWithDoctor finds the nearest hospital at given location and a doctor with his/her expertise, getTransportation finds the transportation related detail, getAppoinment retrieves the appointment detail and getPackage finds the payment related options.

I have added semantic description to these Web services using the ontology developed. These semantically enabled services are stored into the service repository for the discovery, selection and composition. Based on the knowledge-base, I define a user request that results towards the composition solution to fulfill the request.

4.5.2 Ontology Model for Healthcare Information System

The sample ontology presented in Figure 4.5 is the Dentistry Ontology, described using OWL and developed using protégé 4.3. Various subclasses such as Symptoms, Doctor, and

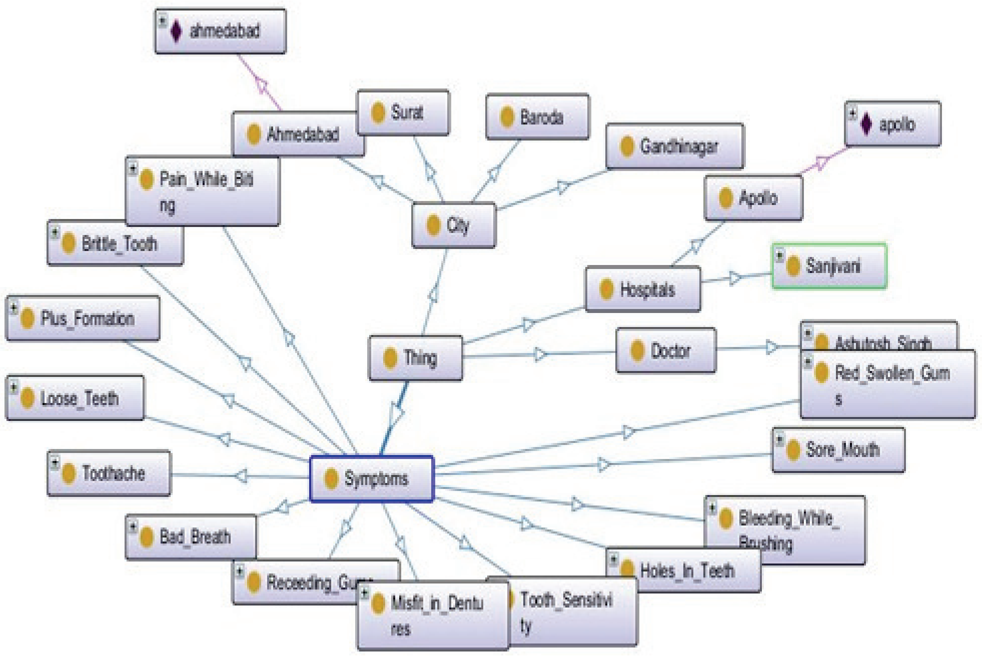


Figure 4.5: A sample of Dentistry Ontology Model

Hospital can be described along with their properties and relationship with each other in the form of concepts. The sample ontology us used as the knowledge base.

4.5.3 Composition Process for Healthcare Information System

The depicts The conceptual representation of dependency relationship among participant medical services to generate a composition plan based on the user input is depicted in Figure 4.6. This composition plan shows that the user will provide the input only once, based on that the composition solution would be generated.

4.5.4 Performance of Healthcare Information System

I have utilized the two QoS constraints, such as response time and throughput as a part of the datasets.

The performance is measured by varying the number of services in the range of 100

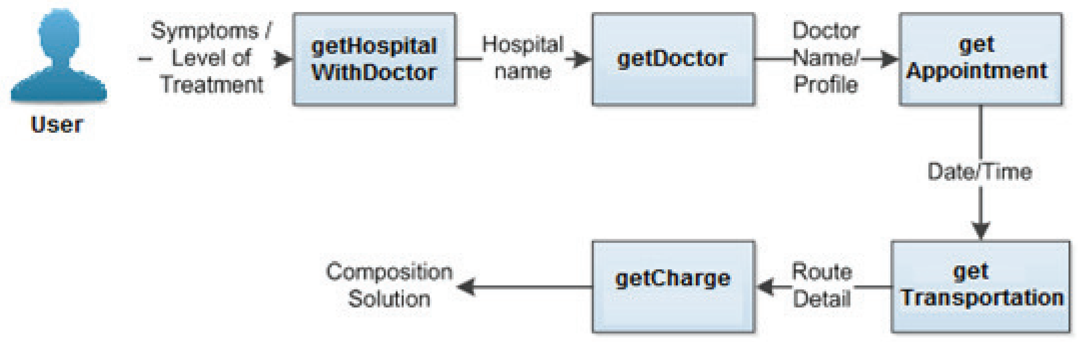


Figure 4.6: Composition plan

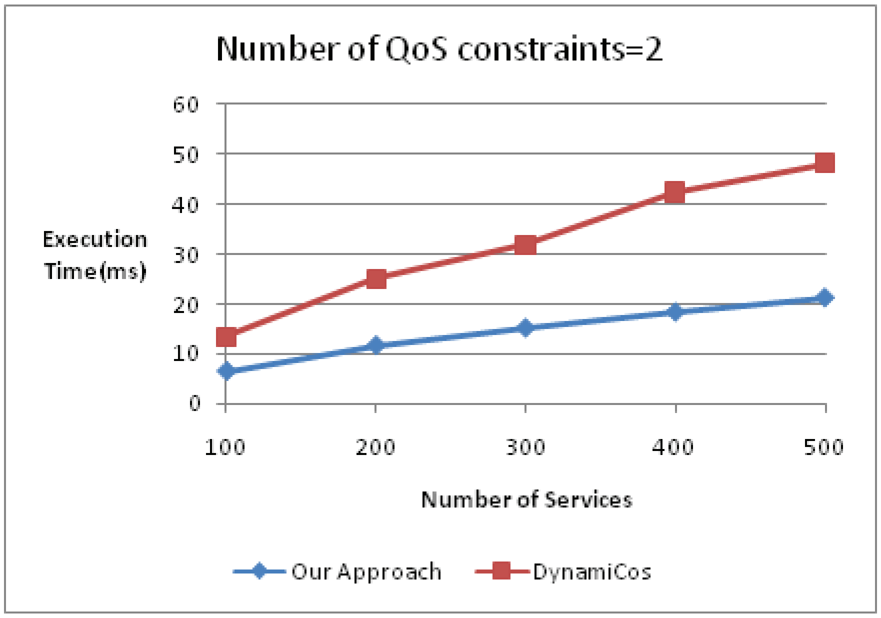


Figure 4.7: Composition process time

upto 500 service for the purpose of comparison with DynamiCoS as shown in Figure 4.7. The obtained measurements show that the execution time of my prototype increases along with the number of services and show improved performance in comparison with DynamiCoS. At this juncture, it is important to note that the DynamiCoS approach has not considered non-functional parameters to perform service discovery, selection and composition tasks.

4.5.5 Comparison with Existing System

The Table 5.5 shows comparison of the proposed prototype with a Healthcare Information System ([Wang et al.) and Hospital application (Da Silva, Pires, and Van Sinderen) developed for medical appointment scenario.

Table 4.2: Comparison with existing prototypes

Aspects	Dynamic Healthcare Service Composition	DynamiCoS	Proposed Prototype
Semantic Web	HL7 Ontology	Domain Ontology	Domain Ontology
Information provided	-Information about the patient within organization -Nurses obtain patientś report -Doctors prescribe treatment	-Provides Hospital location information only - Makes an appointment with doctor	-Provides Hospital Information -Provides appointment with doctor -Provides transportation detail -Offers package detail
Output	-Patientś current report -Treatment given by the doctor	-Appointment detail only	-Appointment schedule -Hospital Detail -Doctor profile - Detail of Treatment charges
Repository-Size	Not specified	500	1000
Comparison	Not given	Not given	Provided
QoS Support	No	No	Yes

4.6 Discussions

In this chapter, I have presented a framework for an integrated approach of service discovery, service selection and service composition for SOAP-based Web services by considering Semantic web and QoS-parameters such as response time and throughput. I have proposed the approach to realize the framework using greedy search technique. The proposed approach has been evaluated using standard WSC datasets and compared with existing approaches of composition. I have demonstrated the application of the approach by developing the prototype for Healthcare Information System by using real Web services. The evaluation work and prototype implementation presents the efficiency and performance of the proposed work.

Chapter 5

RESTful Web services Discovery, Selection and Composition

In this chapter, the proposed framework and approaches for service discovery, selection and composition and experimental work of RESTful Web services have been presented.[1].[2]. Section 5.1 discusses proposed framework for RESTful services discovery, selection and composition; Section 5.2 presents Service Discovery Problem; Section 5.3 presents Service Selection Problem; Section 5.4 presents Web service Composition Problem; Section 5.5 presents Prototype Development for Population Information System; Section 5.6 presents Prototype Development for Healthcare Recommendation System(HRS); Section 5.7 concludes the chapter.

[1]Part of this chapter has been published as Kirit J. Modi, Sanjay Garg,*Discovery and Composition of Link Open Data based RESTful Web services*, International Journal of Computer Science & Communication 6 (2015), 294-304.

[2] Part of this chapter has been communicated as Kirit Modi, Sanjay Garg, Sanjay Chaudhary *An Effective Approach for RESTful Web services using Linked Open Data*, International Journal of Web Information Systems(IJWIS), Emerald Group Publishing Limited, United Kingdom, (Manuscript No.:IJWIS-08-2016-0044) [Submitted on 11/08/2016]

5.1 Proposed Framework for RESTful Web Services Discovery, Selection and Composition

The literature related to the RESTful Web services and Linked Data principles clearly demonstrates that Linked Open Data paradigm is gaining popularity to publish the information as public data, but very less work has been done to make this information easily discoverable and composable. In the following, an effort has been done to provide the solution through a novel framework and approaches based on it. A novel framework for RESTful Web services discovery, selection and composition consists of three main components, i.e. Discovery Engine, Selection Module and Composition Engine.

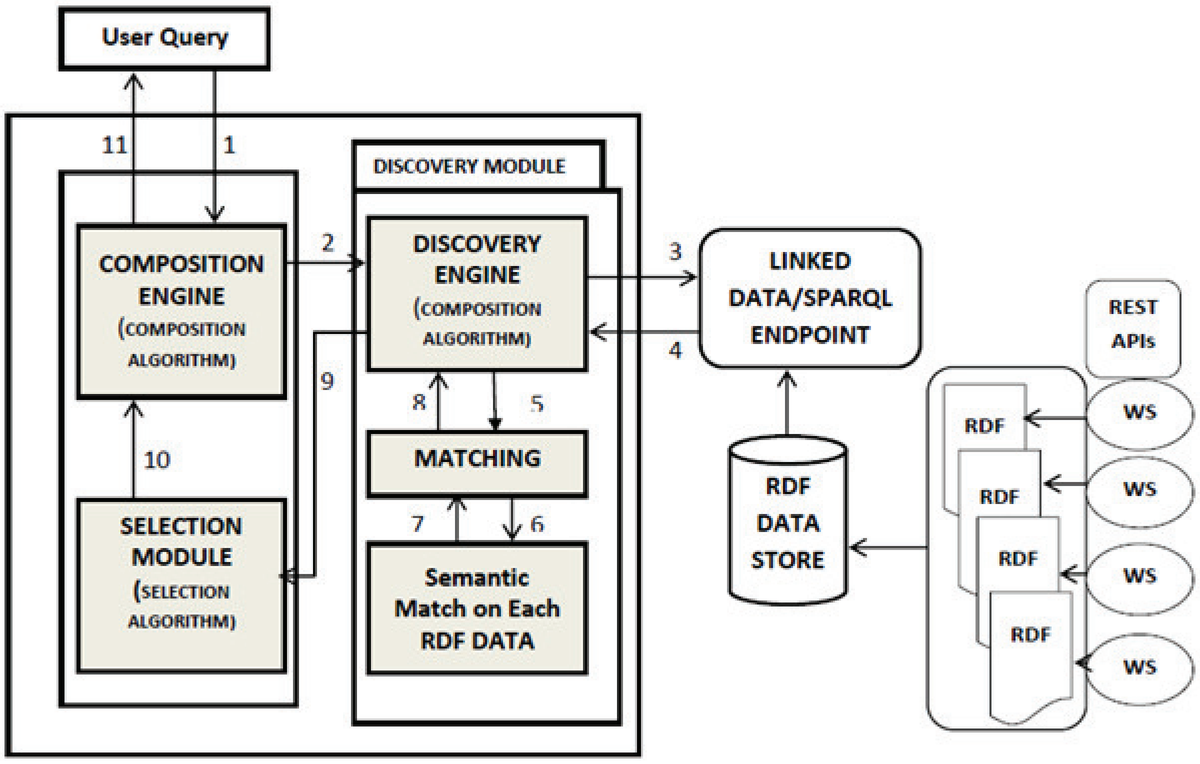

User Query
11
1
DISCOVERY MODULE
COMPOSITION ENGINE (COMPOSITION ALGORITHM)
2
DISCOVERY ENGINE (COMPOSITION ALGORITHM)
3
4
LINKED DATA/SPARQL ENDPOINT
REST APIs
9
8
5
MATCHING
10
7
6
SELECTION MODULE (SELECTION ALGORITHM)
Semantic Match on Each RDF DATA
RDF DATA STORE
RDF
RDF
RDF
RDF
WS
WS
WS
WS

Figure 5.1: RESTful Web Services Discovery, Selection and Composition Framework

Figure 5.1 depicts the components of our approach. The RESTful Web services or REST APIs are collectively placed in the service repository and their semantic descriptions are defined by developing RDF view, which is archived by RDF Data Store. The RDF view is extracted as a set of RDF triples. RDF Data Store is also knows as RDF Triple Store. User requests for resources by specifying required inputs and desired output parameters, which are converted into SPARQL query so that discovery process on RDF Data to be performed. The semantic description of RDF Data is used to perform matching operation by executing SPARQL query which provides the list of base URIs of resources as a result of resource discovery process. These URIs are extracted further, where QoS parameters are used to filter the resources according to the QoS constraints specified by the user which provides filtered resources as a result of selection process. The Composition Engine takes filtered resources as an input in the form of RDF Data on which SPARQL query is applied. As a result of recursive operation, composed solution is generated to satisfy the user need. As per my knowledge, a novel framework is presented by me through integration of discovery, selection and composition processes for RESTful services. These processes are discussed in detail in the following sections one by one.

5.2 Service Discovery Problem

Using the query requested by the user and RDF data stored in the repository, services would be retrieved automatically from the service repository that matches the query requirement. This process is called as the Web services discovery problem.

The Service discovery problem is developed by a set of definitions as below.

- **Dataset (D):** The dataset D contains the multiple RDF files (RDF1, RDF2,..,RDFn), that describes the semantic description of RESTful services.
- **Resultset (Ro):** The Resultset Ro contains matched URIs as per the semantic matching operation.
- **TemporaryResultset (T):** Temporary Resultset T contains each matched RDF data.
- **Query (Si):** It represents input parameters of user request as a SPARQL query.
- **Output (So):** It represents output parameters of user request as a SPARQL query.

Algorithm 4 RESTful Web services Discovery Algorithm

```
Input: $DatasetD = RDF1, RDF2, RDF3, ., RDFn$
$SUQuerySi = QueryRequest(Inputparameter)$
$SUQuerySo = OutputRequest(Outputparameter)$
Output:$ResultsetRo = T1, T2, ..Tn$
Discovery Algorithm(Dataset D, SUQuery Si, SUQuery So, Resultset Ro)
 1: Initialization Ro=   or R=NULL;
 2: if $D =$ then
 3:     $Exit$;
 4: else
 5:     for each $S \in SR$ do
 6:         $InitializationT = orT = NULL$;
 7:         $execute(ExecuteSPARQLquery(RDF \leftarrow SPARQLQuery))$
 8:         if (match (RDF,Si,So)) then
 9:             $T \leftarrow matchedResult$;
10:         end if
11:         if T! =   or T! = Null  then
12:             $ResultsetRo = RoUT$
13:         end if
14:     end for
15:     return $ResultsetR$;;
16: end if
```

Algorithm 5 Match function of RESTful Web Services Discovery Algorithm

Input: SUQuery Si = Query Request (Input parameter) SUQuery So = Output Request (Output parameter)

Output: Service

Match(RDF graph G, Si, So)

```
1: if $(RDF! = null)$ then
2:     $Gs \leftarrow split(G)$;
3:     $Gs1 \leftarrow find(Si, Gs)$;
4:     $Gi \leftarrow find(So, Gs1)$;
5: end if
6: if (Gi!= null) then
7:     $Gi \leftarrow BGiconstructBi(Gi)$;
8: end if
9: $u \leftarrow RS(q, BGi)T$
10: return $Service$;;
```

In the discovery operation of Algorithm 4, based on the user request, each RDF view is evaluated, extracted and stored into the RDF triple store. The variable R is the Resultset, which contains the matched resources retrieved through matching operation initially it is empty (step 1). Termination operation is performed, when triple store is null or Resultset gives an empty set (step 2 to 4). Extract all the RDF Data from the triple store. For each RDF data, we have to match the user request, which is translated into the SPARQL query with RDF Data store. Matched resources generated will be stored into the TemporaryResultset T, which represents the base URI of the matched resource. This TemporaryResultset will be added into the Resultset R till the end of Data (step 6 to 16). In short, I perform searching through semantic matching between the user request and Linked-Data in the form of RDF and if semantic match is found then I extract the base URI of the matched RDF data, which contains the resource information requested by the user.

In the discovery algorithm, step 6 to 16 will execute for each RDF, so the complexity of this algorithm is O(n). Where, n describes the number of RDF data in the dataset.

After successful evaluation of RDF data, match resultset R updates the value of Temporary Resultset T. While reading the next RDF data, TemporaryResultset T will store the current RDF evaluation. The matching function is represented as an Algorithm 5.

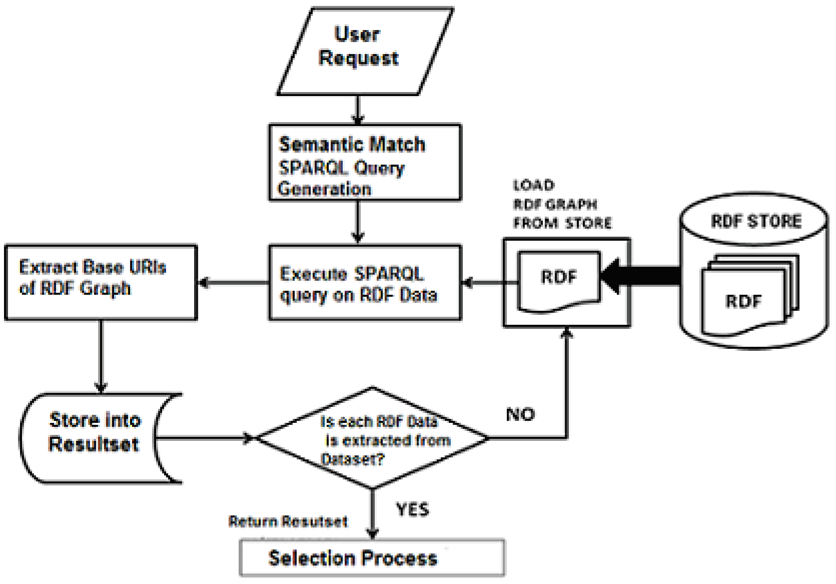

User Request
Semantic Match SPARQL Query Generation
LOAD RDF GRAPH FROM STORE
RDF STORE
Extract Base URIs of RDF Graph
Execute SPARQL query on RDF Data
RDF
RDF
Store into Resultset
Is each RDF Data is extracted from Dataset?
NO
YES
Return Resutset
Selection Process

Figure 5.2: Flowchart of Discovery process

As depicted in the flowchart of discovery process of Figure 5.2, the user requirements are matched with RDF data and if semantic match found then the base URI of the RDF data is extracted, which contains the resources stored in the RDF Data store. The matched resources of discovery process are provided to the selection process for filtering purpose as discussed below.

5.3 Service Selection Problem

After retrieving the discovered services, I need to apply service selection logic to filter out and to assign rank to the services. The QoS Score for service selection can be obtained by using the Simple Additive Weighting (SAW) technique defined in chapter 2. The Web services selection problem is developed by a set of definitions as below.

- **Resultset (Ri):** The Resultset Ri contains a set of matched URIs.
- **Selected Services(SS):**It represents a set of selected services or resources $SS \subseteq SD$.

- **Service Specific QoS Criteria (QD):** It represents service specific QoS criteria to calculate QoS score.
- **Ranked Services (SR):** It represents a set of ranked services . $SR \subseteq SS$.
- **Descending Order of Ranked Services (Qdec)):** It performs the ranking of a set of service or resources in descending order as a set of Ranked services.

Algorithm 6 RESTful services Selection Algorithm

Input::Resultset Ri, Service-QoS SQoS;

Output: Selected web service set SR;

Selection Algorithm (Resultset Ri,SD,SR,)

```
1: $SD \leftarrow Ri$;
2: if $(SQoS = NULL)$ then
3:     $Gs \leftarrow split(G)$;
4: else
5:     $len \leftarrow QD.length$;
6:     $SS \leftarrow SelectWithQoS(SQoS, QD, len)$;
7:     $returnSS$;
8: end if
9: for $i \leftarrow 1toSS$ do
10:     $Qdec = len.value + QD$;
11:     $SR.rank(Qdec)$;
12:     $returnSR$;;
13: end for
```

The service selection algorithm is depicted in Algorithm 6. For the purpose of selection and filtering, user specifies QoS constrains in the request parameters that will be compared with QoS Parameter values of resources (SQoS)(step 1 to 3). A set of matched resources retrieved as result of discovery process will be used as an input to the selection process. For each resource, total number of QoS parameters is identified to calculate the length value (step 5). Based on user specified constraints, resource will be selected (step

6 and 7). Selected resources would be used for assigning rank in descending order based on the threshold specified in the user request. A set of ranked resources will be generated as a result of ranking operation. At this point, I highlight that selection and ranking are key operations to increase user satisfaction still they are mostly neglected by the research community particularly for RESTful services. In the next section, composition process is presented.

5.4 Composition Problem

From the selected set of services or APIs, composition would be resulted by considering a sequence of APIs. A composition problem is developed by defining a set of terms as follows.

- **Resultset (SRi):** The Resultset SRi contains a set of selected resources.
- **Query (Si):** It represents input parameters of user request as a SPARQL query.
- **Output (So):** It represents output parameters of user request as a SPARQL query.
- **Dependency Graph (G):** It represents linking relationship among nodes. The graph contains the linked node, i.e one node depends upon the another node.
- **SparqlResultset (S):** SparqlResultset S is the SPARQL query generated results which contains matching result as per the query requested by the user on each file of R.
- **Temporary Resultset(TSR):** The SPARQL Query have to be executed as per the user requirements and matched results generated from each and every data files, which will store into temporary result set **TSR**.
- **Composed Resultset(C):** Composed resultset C contains the generated composed result.

In the Algorithm 7, I take selected Resultset SR as an input, which contains a set of selected and ranked resources from the service selection process. As per the user request, I have to compose resources for each RDF data stored in the SR by executing SPARQL query. Initially, the values of SPARQL query S and composition values C set to Null

Algorithm 7 RESTful Web Services Composition Algorithm

Input: Resultset Ri = T1, T2, Tn // Matched URIs

SUQuery Si = Query Request (Input parameter)

SUQuery So = Query Request (Output parameter)

Output: SparqlResultset S = Contains the SPARQL query executed results.

G = Dependency Graph

Composition Algorithm: (Resultset R, SUQuery Si, SUQuery So, G)

```
1: Initialization $S = \{\}, C = \{\}$;
2: if $R = \{\}$ then
3:     Exit;
4: end if
5: if $(R! = null)$ then
6:     for each $URIsinResultsetR$ do
7:         for each $Linked - dataR$ do
8:             $GenerateDependencyGraph(G)$
9:             $G \leftarrow Dereferenced(R, G)$;
10:            $TR = RECURSIVEFUNCTION(ExecuteSPARQLQuery(G \leftarrow (SPARQLQuery)))$;
11:            if $(TR! = Null)$ then
12:                S = S U TR;
13:                $S \rightarrow C$;
14:            end if
15:        end for
16:    end for
17: end if
18: return $ResultsetR$;;
```

(step 1). For each base URI of the selected resource as well as for the Linked Dataset, dereferencing of URI is performed by executing SPARQL query on the RDF Data. I utilize recursive approach for evaluating the RDF dataset. As a result, Dependency Graph is generated (step 5 to 17). According to the user request, composition solution is generated which is stored into Compositionset C (step 18). A flowchart of composition process is depicted in Figure 5.3. In the Composition algorithm, each node of RDF is evaluated as

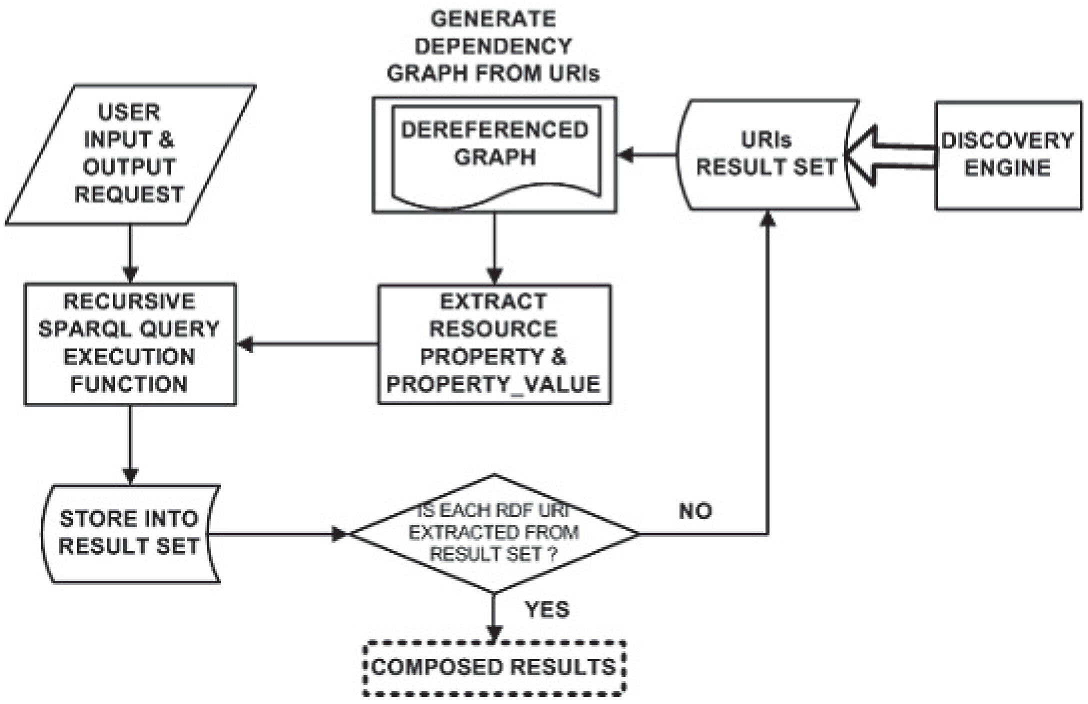

GENERATE DEPENDENCY GRAPH FROM URIs
USER INPUT & OUTPUT REQUEST
DEREFERENCED GRAPH
URIs RESULT SET
DISCOVERY ENGINE
RECURSIVE SPARQL QUERY EXECUTION FUNCTION
EXTRACT RESOURCE PROPERTY & PROPERTY_VALUE
STORE INTO RESULT SET
IS EACH RDF URI EXTRACTED FROM RESULT SET ?
NO
YES
COMPOSED RESULTS

Figure 5.3: Flowchart of Composition process

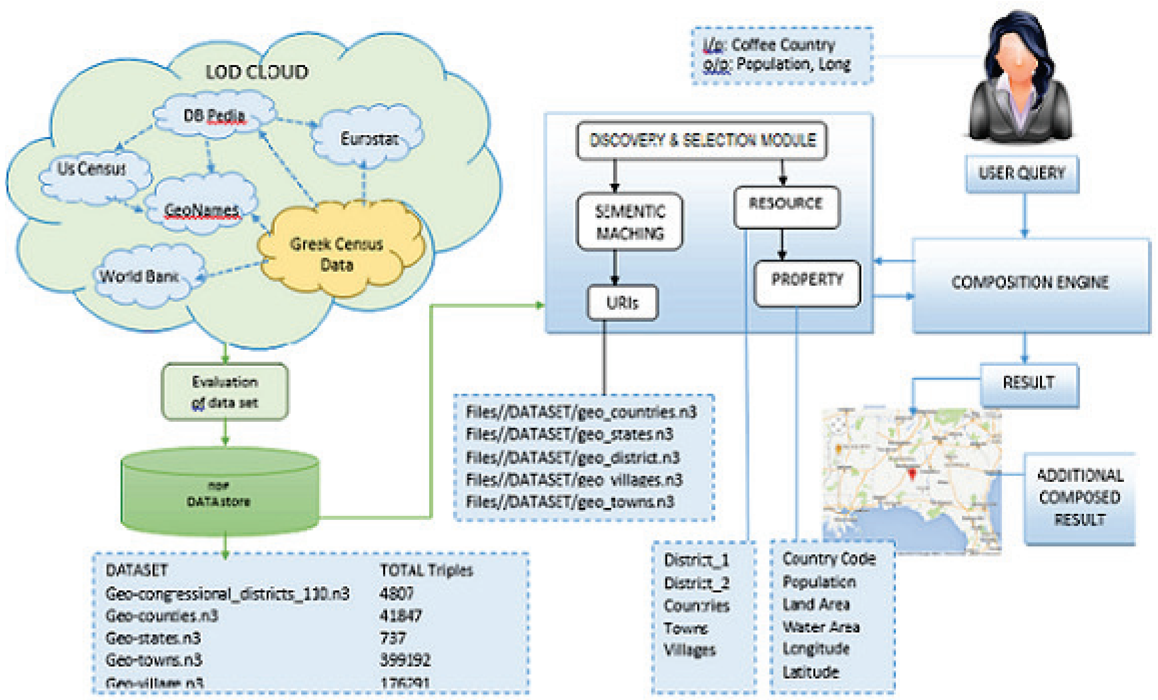
well as its semantic properties are extracted, so the complexity of this algorithm is O(n * m), where n describes the number of RDF data in the dataset and m describes the number of nodes need to be checked for composition. In the following section, I develop prototypes for Population Information System and Healthcare Recommendation System using Linked Data principles and RESTful Web services to evaluate the proposed work.

5.5 Prototype Development for Population Information System

In this section, a prototype for US-based Population Information System is developed to demonstrate the efficiency and effectiveness of the proposed work.

5.5.1 Prototype Framework for Population Information System

A prototype framework for Population Information System (PIS) using Linked Open geospatial data is depicted in Figure 5.4. In the prototype development, I have considered the U.S. Census dataset (Tauberer), available as Linked Open Data that contains over a billion RDF triples and describes the rich information about the population statistics at different geographic levels, from the U.S. as a whole, down through states, counties, sub-counties.



DATASET	TOTAL Triples
Geo-congressional_districts_110.n3	4807
Geo-counties.n3	41847
Geo-states.n3	737
Geo-towns.n3	399192
Geo-village.n3	176291

Files//DATASET/geo_countries.n3
Files//DATASET/geo_states.n3
Files//DATASET/geo_district.n3
Files//DATASET/geo villages.n3
Files//DATASET/geo_towns.n3

District_1
District_2
Countries
Towns
Villages

Country Code
Population
Land Area
Water Area
Longitude
Latitude

Figure 5.4: Prototype Framework for US-based Population Information System

As per the RDF format, conceptual Resource, Property and Property_values are derived from the U.S. census dataset. As shown in Figure, Userś query(input and output parameters) is provided to the Discovery engine for searching the requested services through

Composition Engine. For example, our input query is Coffee_country and output query is Population and Longitude.

Discovery engine performs semantic match the user query (i/p; o/p) and the RDF data that retrieved from data store. As a result, discovery engine provides the list of base URIs list of resources.

The retrieved URIs from the discovery process results are provided to the Selection Engine which performs the selection and filtering of matched resources. The filtered resources are passed to the Composition Engine, which generates the composed result. As per our earlier example the composed result is Population = 43615 and Longitude=-85.956841. Moreover, additional results related to that county like Country_code, Longitude, Latitudes, Land_area, etc are generated through discovery process.

5.5.2 Experimental Setup

For the experiments, the setup is configured with specification as follows: Operating System :Windows XP, Framework: Microsoft .Net Framework 4.0, RAM :2.5 GB, Software Tool: Microsoft Visual Studio 2010, Processor: Intel Core 2 Duo, Space Requirement:5 GB.
The prototype is implemented using .Net framework 4.0 and Open Source .Net Library (Vesse et al.) on .Net platform, which provides a convenient environment to work with RDF, and SPARQL query. dotNetRDF (Vesse et al.) provides toolkit for the user which in turn provides the way to work with single RDF file, which contains RDF Editor, SparqlGUI and so on. It also provides an open source SemWeb.NET library for programming purpose.

The objective to setup this experimental configuration is to expose the libraries and tools related to Semantic Web programming. The presented platform found to be more suitable to implement the Semantic RESTful Web services using Lined Open Data.

5.5.3 RDF Data Model

The Data model contains the data in RDF/XML format which contains linked URIs, i.e. One URI is depend upon the other its relevant URI. Using these URIs, RDF graph has been generated. In the present scenario of US census dataset, I get the following URIs:

- http://www.rdfabout.com/rdf/usgov/geo/us/al/counties/autauga_county/autaugaville/autaugaville (geo_village: Autaugaville).
- http://www.rdfabout.com/rdf/usgov/geo/us/al/counties/autauga_county/billingsley/billingsley (geo_village: Billingsley).
- http://www.rdfabout.com/rdf/usgov/geo/us/al/counties/bibbcounty (geo_county: Bibb).
- http://www.rdfabout.com/rdf/usgov/geo/us/vt/cd/110/1 (geo_dist: Cong_dist1).

The two geo_villages dc: Autaugaville and dc: Billingsley are connected to geo_towns dc: Autaugaville and dc: Billingsley respectively.

5.5.4 Experimental Work

This section focuses on detail about performance parameters, dataset and various steps to be performed to show the applicability of the proposed work.

(i.) **Performance Parameters**

Following performance parameters have been considered to measure the performance of the approach. Based on the literature related to RESTful Web services, I found following two parameters as the most appropriate to measure the feasibility and usability of the proposed approach.

(a.) **Number of Services:** It defines total number of services available in the repository. It measures the scalability of the approach.

(b.) **Execution Time:** It represents the time required to perform the composition process.

5.5.5 Service Discovery Process

The resources modeled in the scenario and their corresponding URIs are presented in Figure 5.5. The particular resources are stored in their respective repositories (districts, countries, states, town, and village) and have their standard URI patterns. Base URIs of resources are retrieved by performing semantic matchmaking in the discovery process.

5.5.6 RDF Data Extraction

The results shown in Table 5.1 are extracted from the U.S. Census dataset, which represents RDF triple store in the form of subject, predicate and object. Based on that conceptual Resource, Property and Property_value, results are derived for each RDF data of U.S. Census dataset.

- Districts:DATASET\geo—congressional_districts_ 110.n3
- Countries: DATASET\geo—counties.n3
- States: DATASET\geo—states.n3
- Town: DATASET\geo—towns.n3
- Village: DATASET\geo—villages.n3

Figure 5.5: Base URIs of discovered resources

Table 5.1: Extracted Results from U.S. Census dataset

RDF Data	Total Triples	Subject Nodes	Predicate Nodes	Objects Nodes	Extraction Time (ms)
geo-congressional_ districts_110.n3	**4807**	**489**	**11**	**3335**	**102**
geo-counties.n3	**41847**	**3271**	**13**	**30894**	**281**
geo-states.n3	**737**	**53**	**14**	**610**	**30**
geo-towns.n3	**399192**	**39438**	**11**	**247665**	**640**
geo-villages.n3	**176291**	**26256**	**11**	**116138**	**542**
TOTAL	**622874**	**69507**	**60**	**398642**	**1595**

5.5.7 Service Selection Process

Performance evaluation of the selection process is shown in Table 5.2, which provides the number of selected services with their QoS values of Throughput and Response Time.

Table 5.2: Result of selection process

Dataset	Total service	No. of requests	No.of selected services	Through-put	Response time (sec.)
1	107	10	4	2.5215	4.2
		25	16	4.7457	5.92
		50	13	11.4821	3.86
2	321	10	5	4.6834	4.37
		25	17	13.4832	3.43
		50	11	10.2571	5.10

5.5.8 Service Composition Process

The selected resources through selection operation will need to be used to generate the composition plan as shown in Figure 5.6, which represents the relationship among various participant resources. Final composition result is visualized using Google map service as presented in Figure 5.7.

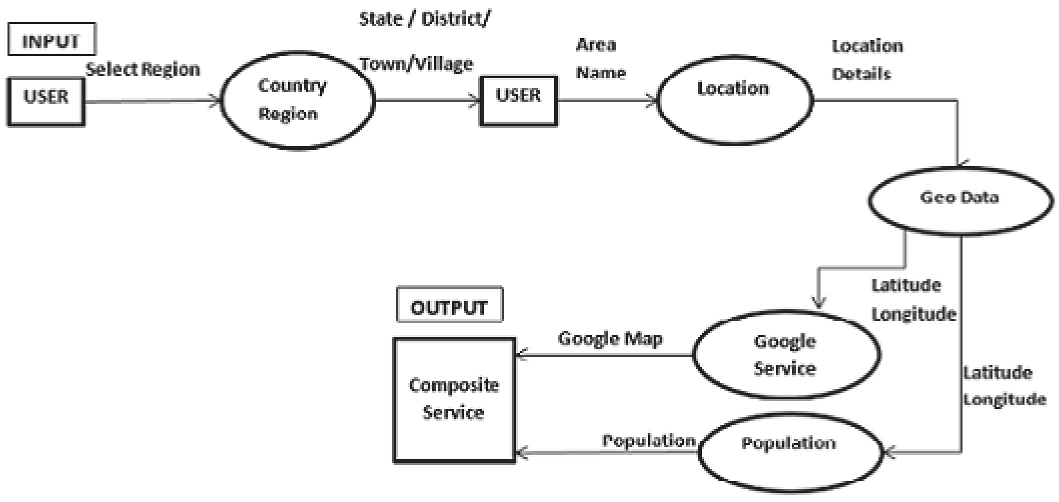


Figure 5.6: Population service composition plan

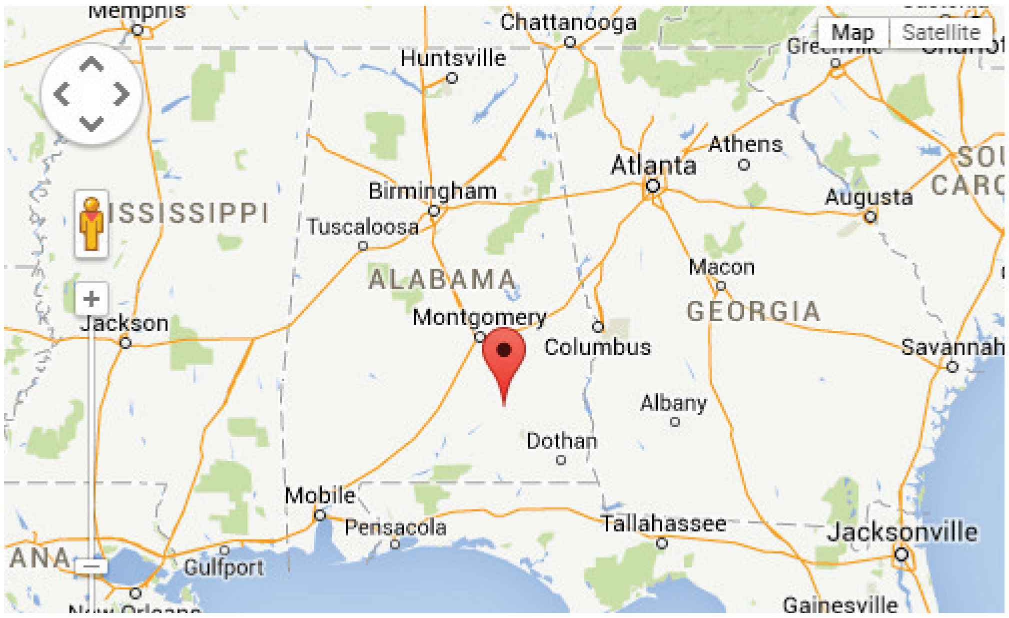


Figure 5.7: Visualization of Composition result

5.5.9 Performance Evaluation

In this section, the number of resources discovered, selected and composed are shown for different values of user input. 5.3.

Table 5.3: Performance evaluation of service discovery, selection and composition

User Input	**Output**	**Total No. of services**	**Discovered services (URI)**	**Ex. Time (ms)**	**Selected services**	**Ex. Time (ms)**	**Composed services**	**Ex. Time (ms)**
Lat.,Long., Popul. for Congre.Dist.	**Lat= 37.61, Long=-122.39, Popul.=639088**	**20000+**	**4031**	**112**	**1094**	**996**	**143**	**1136**
Lat.,Long., Population, Land area for Butler county	**Lat= 31.73, Long=-86.66, Popul.= 21399, Land= 20120706972m^2**	**20000+**	**5578**	**30**	**1201**	**815**	**104**	**903**

5.5.10 Comparison with Existing Approaches

I have considered the work proposed by Yong-Ju Lee et al.(Lee) and Bohara et al.(Bohara, Mishra, and Chaudhary) for the comparison with our proposed work. As a prototype, I have identified Population Information System based on Linked Open Data while Lee el al. demonstrated Traveling Information System and Bohara et al. demonstrated Crop Recommendation System, both of them are based on Domain Ontology for semantic description. I have focused on automatic approach in sense that user involvement is avoided during the process while Lee et al. and Bohara et al. requires user interaction to complete the process in that sense they are manual in nature.

5.6 Prototype Development for Healthcare Recommendation system

In this section, a prototype for Healthcare Recommendation System is developed using Linked Open Data and RESTful Web services by me to demonstrate the efficiency and effectiveness of the proposed work.

5.6.1 Prototype Framework for Healthcare Recommendation system

A prototype framework for Healthcare Recommendation System using Dataset of H1N1 is presented in Figure 5.8.. The prototype manages the patientś details, emergency transportation, symptoms information, recommended solution, affected areas, and location of resource centre and so on.

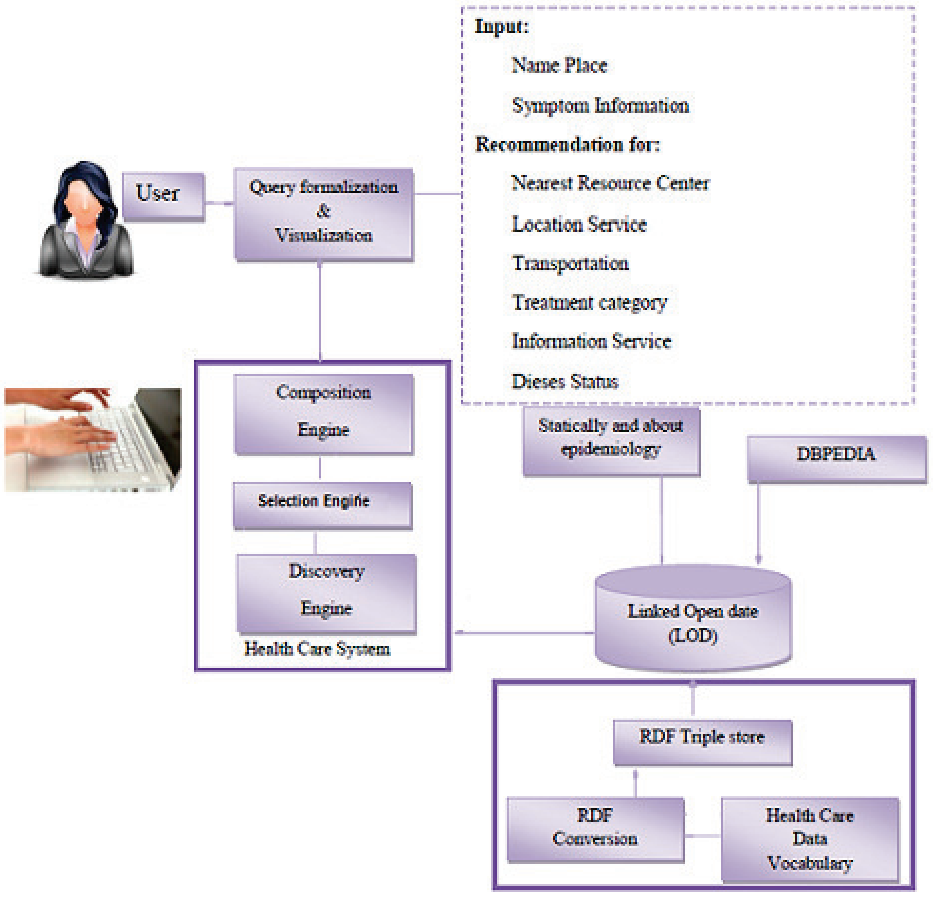


Figure 5.8: Prototype Framework for Emergency Healthcare Recommendation system

The components of the prototype are: Composition Engine, Discovery Engine, RDF Processing Engine and Linked Open Data. The main functionality of the RDF Processing Engine is to convert locally stored medical data into RDF Data and link them with LOD. The RDF Conversion Process is used for representing medical data as RDF graphs with the help of existing conversion tools such as triples. Due to the limited availability of existing vocabularies to define concepts and their relationships in the context of healthcare information, Healthcare Data Vocabulary is developed. To store the generated RDF triples through the RDF conversion process, RDF repository is needed while LOD Publisher is the server component for publishing the RDF triples and LOD to make them available as RDF dumps for other interested parties. RDF repository and LOD publisher are not shown in the framework. In order to support a proper decision making process and to trigger actions corresponding to a userś request, a specific set of decision rules need to be defined as a part of discovery, selection and composition process.

5.6.2 Experimental Setup

For the implementation, following specification is used: Operating System: Windows XP, RAM: 2.5 GB, Software Tool: Eclipse Europa , Processor: Intel Core 2 Duo, Space Requirement:5 GB. This prototype is implemented using Jena API on Java platform which provides an easy and powerful mechanism to with RDF, SPARQL. Protégé tool provides support for the ontology development which in turn provides the way to work with RDF.

5.6.3 Dataset

In the Experimental work, I have considered the healthcare dataset which describes rich information about the Web services with semantic knowledge concept. This dataset describes setting up the dataset as Linked Data which contains many triples related to healthcare system as RDF format. A prototype for Healthcare system has been developed using RESTful services and Linked Open Data. For the prototype, data of H1N1 disease during the period from 2009 to 213 has been taken, which contains number of cases and mortality rate of various states of India as shown in Figure 5.9 ("[Cases of Influenza A H1N1 (Swine Flu) State/UT wise, Year wise for 2009, 2010, 2011 and 2012"),("Laboratory confirmed Cases and Deaths caused by Pandemic Influenza A H1N1: State/ UTwise”).

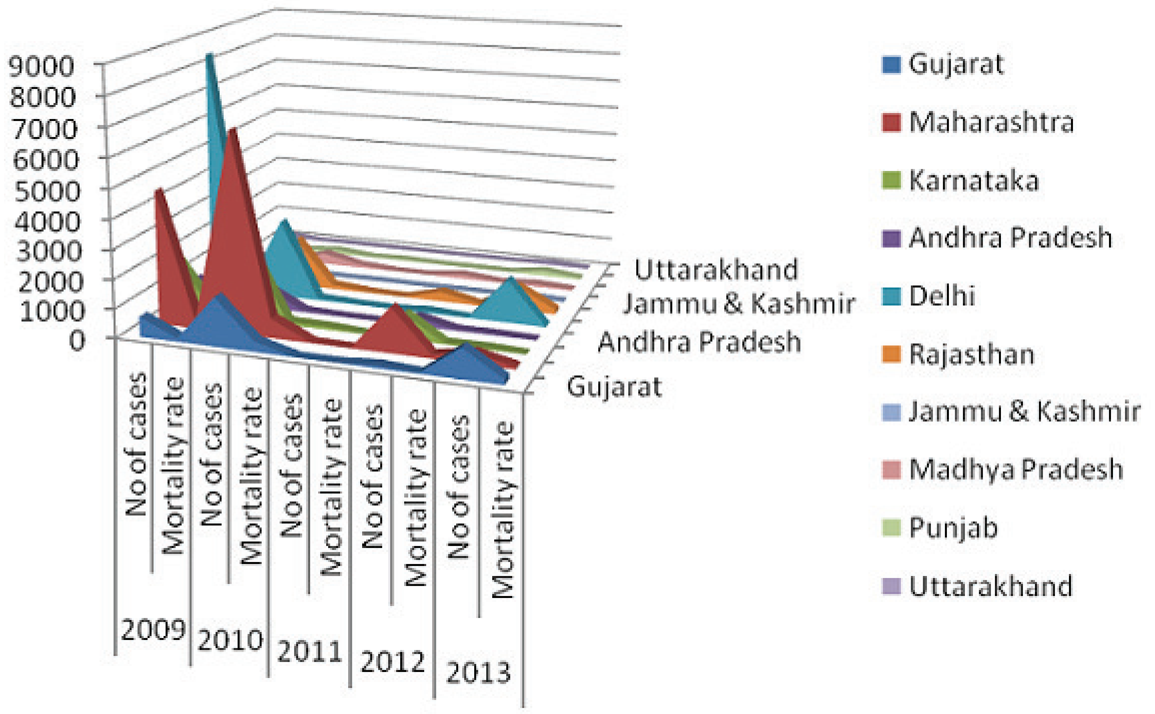


Figure 5.9: H1N1 Data of No. of cases and Mortality rate 2009-2013 in India

5.6.4 RDF Data Model

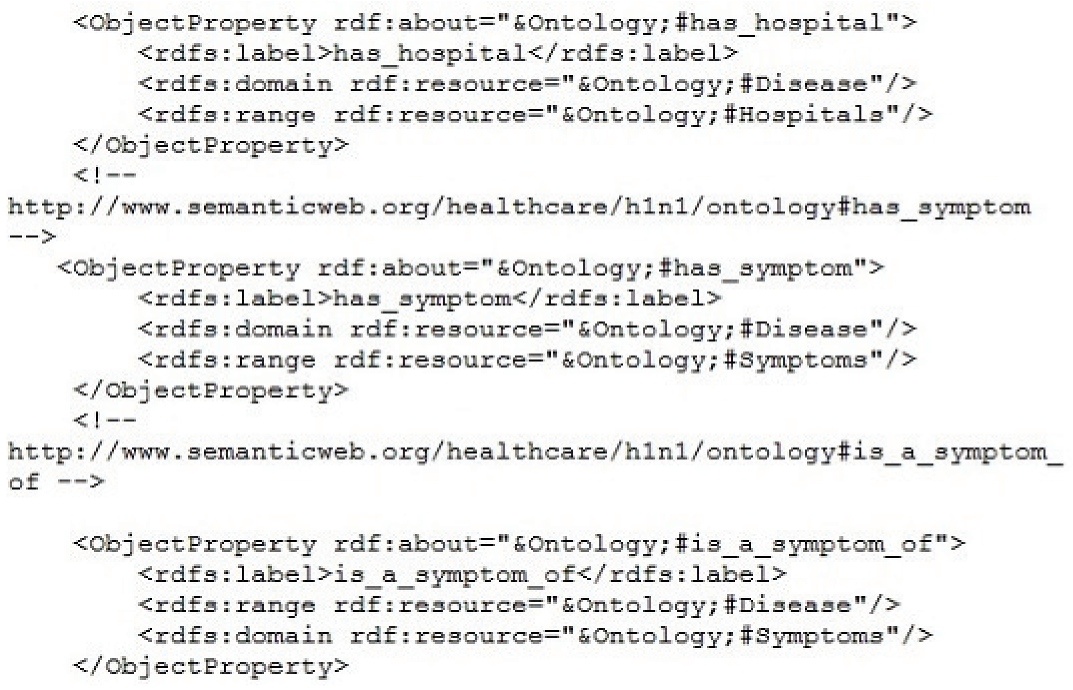
Dataset contains following file in RDF/XML format which contains linked data concepts as depicted in figure 5.10.

5.6.5 Experimental Results

This section focuses on detail about dataset and various steps to be performed to show the applicability of the proposed work.

5.6.6 Service Discovery Process

During the discovery process, resources have been modeled as RDF data and retrieved corresponding URIs of the RDF data based on the SPARQL query execution. The particular resources are stored in their respective repositories and have the URI patterns as shown in Figure 5.11. I have considered resources, such as symptoms, disease, hospitals and transportation from the ontology model developed by us and location resource from the



```
    <ObjectProperty rdf:about="&Ontology;#has_hospital">
        <rdfs:label>has_hospital</rdfs:label>
        <rdfs:domain rdf:resource="&Ontology;#Disease"/>
        <rdfs:range rdf:resource="&Ontology;#Hospitals"/>
    </ObjectProperty>
    <!--
http://www.semanticweb.org/healthcare/h1n1/ontology#has_symptom
-->
   <ObjectProperty rdf:about="&Ontology;#has_symptom">
        <rdfs:label>has_symptom</rdfs:label>
        <rdfs:domain rdf:resource="&Ontology;#Disease"/>
        <rdfs:range rdf:resource="&Ontology;#Symptoms"/>
    </ObjectProperty>
    <!--
http://www.semanticweb.org/healthcare/h1n1/ontology#is_a_symptom_
of -->

    <ObjectProperty rdf:about="&Ontology;#is_a_symptom_of">
        <rdfs:label>is_a_symptom_of</rdfs:label>
        <rdfs:range rdf:resource="&Ontology;#Disease"/>
        <rdfs:domain rdf:resource="&Ontology;#Symptoms"/>
    </ObjectProperty>
```

Figure 5.10: RDF/XML data

LoD cloud dbpedia. The presented Base URIs are retrieved as a result of discovery process.

- http://www.semanticweb.org/healthcare/h1n1/ontology/symtoms
- http://www.semanticweb.org/healthcare/h1n1/ontology/dieses
- http://www.semanticweb.org/healthcare/h1n1/ontology/hospitals
- http://www.semanticweb.org/healthcare/h1n1/ontology/trasportation
- http://dbpedia.org/ontology/location

Figure 5.11: Base URIs retrieved through discovery process

5.6.7 RDF Data Extraction

- <http://www.semanticweb.org/healthcare/h1n1/ontology#Body_ache>
- <http://www.semanticweb.org/healthcare/h1n1/ontology#Strep_Throat>
- <http://www.semanticweb.org/healthcare/h1n1/ontology#Cough>
- <http://www.semanticweb.org/healthcare/h1n1/ontology#Throat_Pain>
- <http://www.semanticweb.org/healthcare/h1n1/ontology#Fatigue>
- <http://www.semanticweb.org/healthcare/h1n1/ontology#Chest_Pain>
- <http://www.semanticweb.org/healthcare/h1n1/ontology#Sore_throat>
- <http://www.semanticweb.org/healthcare/h1n1/ontology#Headache>
- <http://www.semanticweb.org/healthcare/h1n1/ontology#Fever>
- <http://www.semanticweb.org/healthcare/h1n1/ontology#Chills>

Figure 5.12: RDF representation of symptoms, diseases

The results shown in figure 5.12 are extracted from the health care system dataset; total number of nodes is extracted in the triple format of semantic linked relation. Based on that conceptual Resource, Property and Property value, results are derived for each RDF data in healthcare system dataset. Base URIs of resources are retrieved by performing semantic matchmaking operation in the discovery operation.

5.6.8 Service Selection Process

The Performance evaluation of the selection process is shown in Table 5.4, which provides value of the selected services with their QoS values of Throughput and Response Time.

Table 5.4: Result of selection process

Dataset	Total services	No. of requests	No.of selected services	Through put	Response time (sec.)
1	128	20	3	3.7735	5.3
		40	10	5.8394	6.85
		60	8	13.6674	4.39
2	481	20	8	5.4794	3.65
		40	11	15.0943	2.65
		60	8	13.363	4.49

5.6.9 Service Composition Process

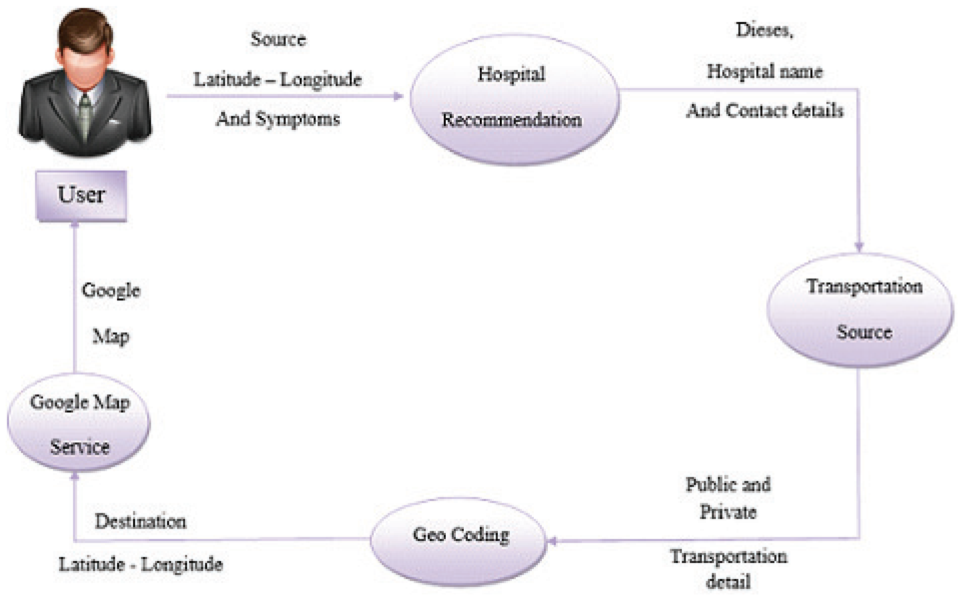


Figure 5.13: work-flow plan for Healthcare Recommendation System

The Figure 5.13 shows relationship among various participant services. From userś input to the composition result, various services like Hospital Recommendation, Transportation, Geo Coding will be invoked by passing parameters from one service to other.

In this conceptual representation of the composition plan, user has to provide input only once. Based on that, discovery and selection tasks will identify the a set of participant services, which will generate composition plan automatically using the composition algorithm proposed here.

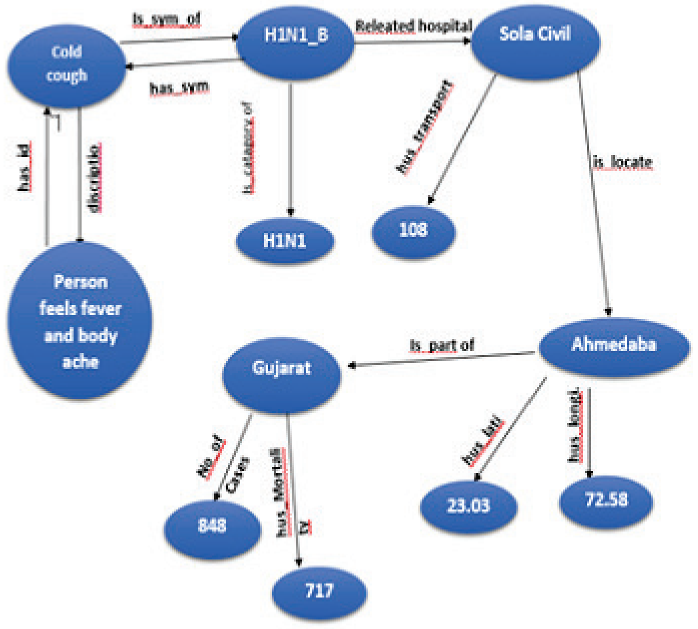


Figure 5.14: RDF graph for Healthcare Recommendation System

As a result of composition process, RDF graph is generated. A RDF graph of Healthcare Recommendation System is shown in Figure 5.14, where user enters symptoms (e.g. Cold cough) and nearest location (e.g. Ahmedabad), Healthcare Recommendation System will process symptoms related disease and disease related hospital. As a result, hospital information will be retrieved first, then transportation details to reach the hospital would be provided.

In this graph, symptoms related disease is H1N1_B category and disease related hospital is sola_civil according to users request. Various services will be found many like Hospital, Transportation, Disease etc. Composition process will be performed on the retrieved services. The Figure 5.15 depicts LOD cloud model to interlink Linked data with cloud. Through DBpedia, it becomes possible to link the LOD and make data publicly accessible. Symptoms & disease are input of the hospital and RDF data, RDF data will be

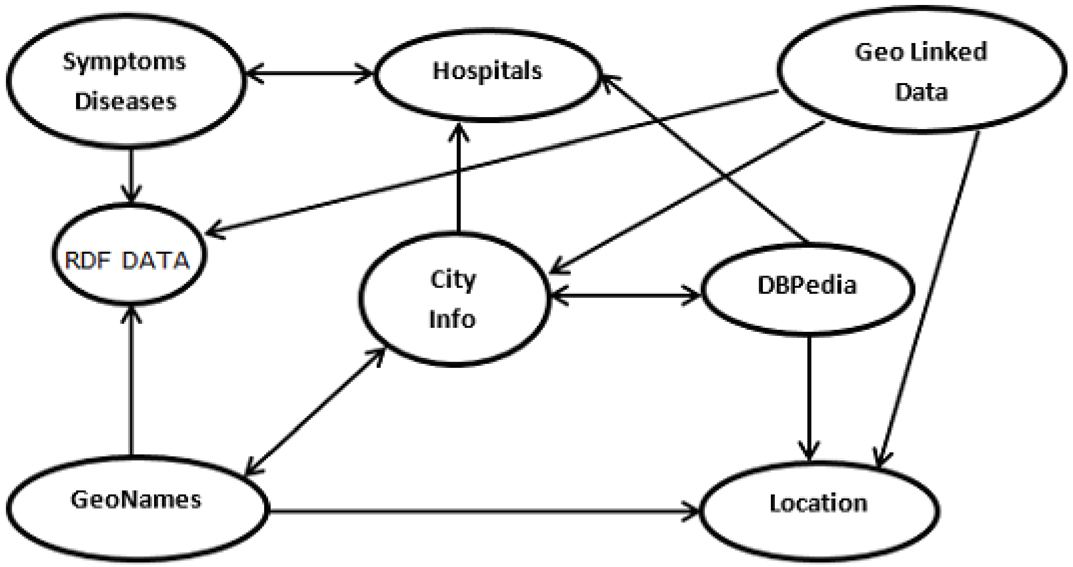
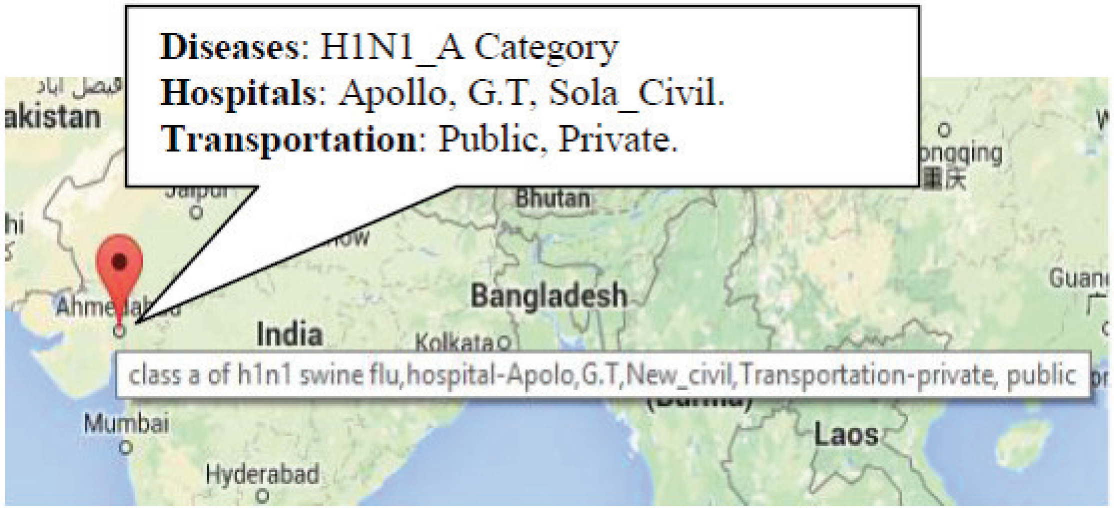


Figure 5.15: LOD cloud Model for Healthcare Recommendation System

a GEOLinked data and GEOnames.

Based on the userś query, recommended composed solution is presented in Figure 5.16, which provides result of Disease category, Hospital Name, Transportation detail etc.

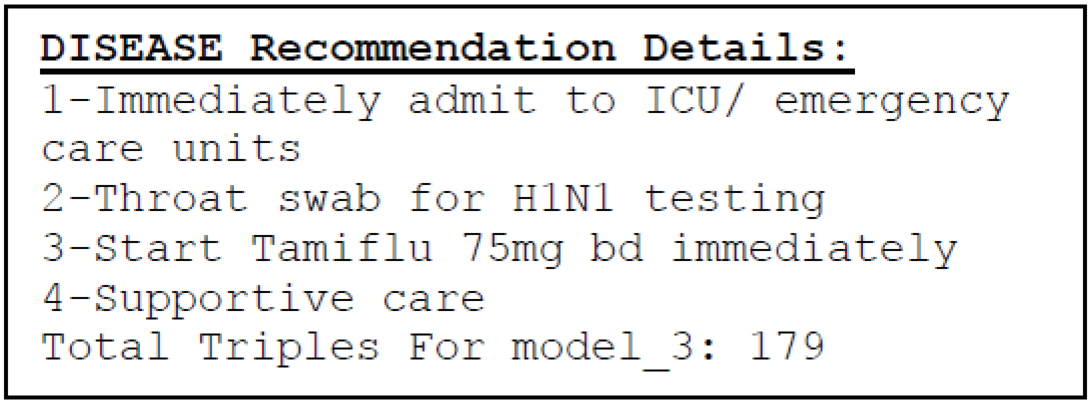


Diseases: H1N1_A Category
Hospitals: Apollo, G.T, Sola_Civil.
Transportation: Public, Private.

class a of h1n1 swine flu,hospital-Apolo,G.T,New_civil,Transportation-private, public

Figure 5.16: Visualization of recommended composed result

Treatment recommendation is also provided to the user based on the Disease category identified based on the specified symptoms by the user as shown in Figure 5.17.



```
DISEASE Recommendation Details:
1-Immediately admit to ICU/ emergency
care units
2-Throat swab for H1N1 testing
3-Start Tamiflu 75mg bd immediately
4-Supportive care
Total Triples For model_3: 179
```

Figure 5.17: Recommendation Detail

5.6.10 Comparison with existing prototypes

A comparison of proposed work with existing prototypes, which are based on Linked Open Data and Domain Ontology are presented in Table 5.5. For the comparison purpose, I have considered parameters, such as objective, Ontology model, Dataset, Information about and Output of the prototype. I have observed that these existing prototypes have focused on data representation and visualization process. Along with above features, I have considered the searching, selection and composition process using LOD based RDF data.

Table 5.5: Comparison with existing prototypes

Aspects	**Bukhari (Bukhari and Baker)**	**Tilahun (Tilahun et al.)**	**Proposed Prototype**
Objective	Canadian health census is translated from its native raw format to LOD, & supports users to extract information from it.	Linked Open Data are utilized for representing, visualizing, retrieving Healthcare information	Enables users to query and to get recommendation of epidemiology i.e.H1N1 using Open Healthcare data
Ontology	HL7 Ontology	Domain Ontology	Dental domain Ontology
Dataset	LOD + Ontology Link with LOD Cloud	LOD	LOD + Ontology Link with LOD Cloud
Query	SPARQL	SPARQL	SPARQL
Informa-tion About	Breast cancer survival. Breast feeding mothers. Kids physical activity Overall health indicator Smoking practice	Trend of HIV prevalence. Location service	Status of H1N1. Transportation & location detail. Treatment detail. Awareness Information
Output	Show the year. No. of patients for specific Diseases & their survival report for both genders with their age. Show the no. of deaths year wise. Display the year & location.	Different kinds of visualizations	Information on map: Location of health centre. Year wise & state wise, no. of positive cases & mortality rate. Recommended treatment & follow up

5.7 Discussions

In this chapter, I have presented a framework for service discovery, service selection and service composition for RESTful Web services by considering Linked Open Data (LOD) and QoS-parameters, such as response time and throughput for service selection and composition. I have proposed the approach to realize the framework. The approach has been evaluated by developing the prototypes for Population Information System and Healthcare Recommendation System based on publicly available Linked data of US Census data and development of Linked data for H1N1 from the government published information respectively. I have compared the proposed system with existing systems to demonstrate the efficiency and effectiveness of the proposed work. As per my knowledge, this is the novel work proposed by me to present the approach for RESTful services by considering service discovery, selection and composition processes in an integrated manner.

Chapter 6

Cloud services Discovery, Selection and Composition

In this chapter, the proposed framework and approaches for service discovery, selection and composition and experimental work of Cloud services have been presented. [1]. Section 6.1 discusses proposed framework for Cloud services discovery, selection and composition; Section 6.2 presents Problem formulation; Section 6.3 presents Experimental work and Results; Section 6.4 presents Prototype for Healthcare Decision Making System; Section 6.5 concludes the chapter.

6.1 Proposed Architecture for Cloud service Discovery, Selection and Composition

The proposed framework is presented in Figure 6.1, for cloud service matchmaking, selection and composition.

As shown in Figure 6.1, a process begins when user requests some functionality from the cloud service by sending request to the broker. User provides input parameters, required output and Quality of Service (QoS) constraints in the request. The matchmaking

[1]Part of this chapter has been communicated as Kirit Modi, Sanjay Garg, Sanjay Chaudhary *Cloud Service Matchmaking, Selection and Composition Approach using QoS and Semantic Web*,Applied Computing and Informatics, Elsevier, (Manuscript No.:ACI-D-16-00123R2)

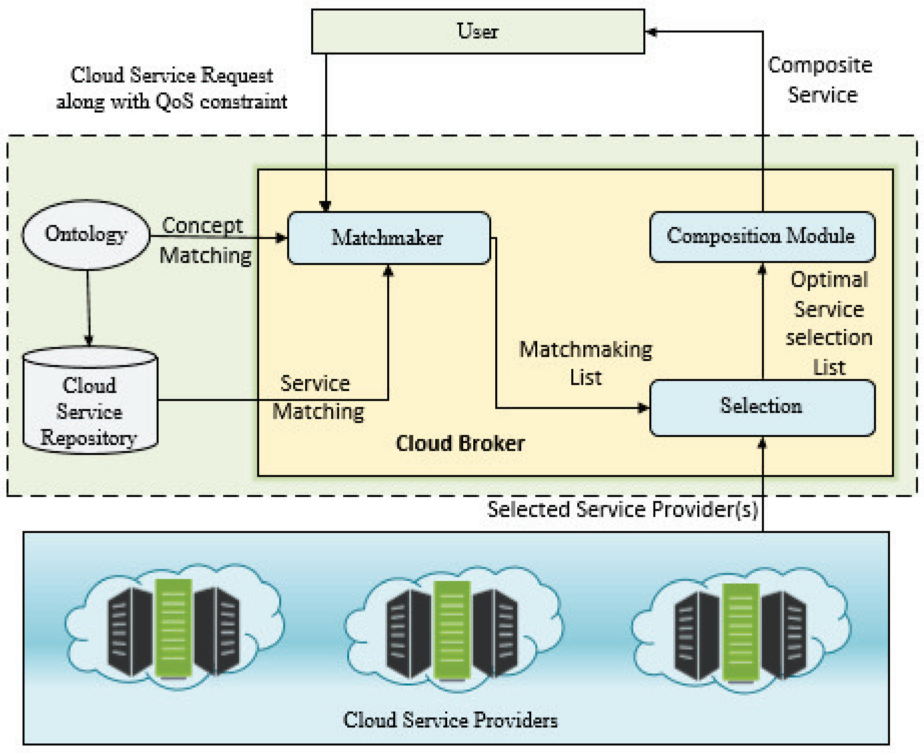


Figure 6.1: Proposed framework for cloud Service matchmaking, selection, and composition

engine starts to discover a set of services those provides required output and fulfills QoS constraints imposed by the user through matchmaking process between the user request and the services stored in the cloud service repository. The matchmaking process is performed with the help of domain ontology. The matchmaker provides a set of matched services to the service selection and filtering module. QoS parameters, i.e. availability, throughput and response time are used here for the selection and filtering of the matched services. Cloud service composition module will use the ranked service to generate the composition plan for the cloud providers. Cloud service provider connected with cloud service repository to get updates the cloud service repository.

The proposed framework consists of following components.

- **User (Service Requester):** Service requester is a computer or a person who want to use cloud services. They provide functionality along with QoS constraints as input to get the desired service provider as output.

- **Cloud Broker:** It is an intermediate entity between the Cloud service providers and users to facilitate functionalities like, publication, negotiation, discovery, selection and integration of services

- **Composition Module:** It is responsible to compose service providers. The output of this module can be a composition plan, or a reference to a composite solution.

- **Selection:** Service selection is responsible to select and filter a set of matched services. After selection, services are filtered based on requesters QoS constraints.

- **Matchmaker:** The semantic matchmaker is used to perfrom matachmaking between user request and a set of services available in the cloud service repository. If the service is matched then it would be added to the candidate list to develop a set of matched services.

- **Ontology:** The ontology act as a knowledge-base to perform semantic matchmaking and to describe semantic description of Cloud services.

- **Cloud Service Repository:** This repository contains the semantically annotated Cloud services. This repository is connected with the matchmaking module, so that Cloud services can be discovered from the repository. QoS information is associated with the Cloud services.

- **Cloud Service Providers:** It is a set of service providers to provide cloud services which represents Infrastructure part of multi-cloud environment.

A cloud service matchmaking, selection, and composition approach can be developed by defining the basic terminology as in the following subsections.

6.2 Cloud service matchmaking

A cloud service matchmaking problem can be described by using notations and algorithms as follows:

- $(R_{VM}$ **(Requested Cost of virtual machine):** a requested value of cost of virtual machine. The cost of virtual machine is predefined by the user for different virtual machines.
- $(R_{AV}$ **(Requested availability):** a parameter demanded by user to the cloud service broker for fulfillment of availability of Cloud services.
- $(R_{TH}$ **(Requested throughput):** a parameter demanded by user to the Cloud service broker for fulfillment of throughput requirement from the Cloud services.
- $(R_{RT}$**(Requested response time):** a parameter demanded by user to the cloud service broker for fulfillment of response time requirement.
- $(P_{VM}$ **PVM (Provided Cost of virtual machine):** a provided value of cost of Virtual machine. The cost of virtual machine is defined by the Cloud Service Provider.
- $(P_{AV}$ **(Providers availability):** a parameter to represent the Providers availability which is defined by the Cloud Service Provider.
- $(P_{TH}$ **(Providers throughput):** a parameter to represent the Providers throughput which is defined by the Cloud Service Provider.
- $(P_{RT}$ **(Providers response time):** a parameter to represent the Providers response time which is defined by the Cloud Service Provider.
- $Sem\_(S_LIST)$ **(Service list):** semantically enabled service list of requested multi-Cloud services.
- $Sem\_(P_LIST)$**(Provider list):** semantically enabled provider list of candidate IaaS providers.
- **CloudMatchmakingList (Cloud Matchmaking list)**: a list of providers that matches the requirements specified by the user.
- $(W_{CST}$ **(Weight of the Cost):** the weight assigned by user to the cost requirement. This specifies the priority to cost requirement.

- (W_{AV} **(Weight of the Availability):** the weight assigned by user to the availability requirement. This specifies the priority to the QoS requirement availability.
- (W_{RT} **(Weight of the Response Time):** the weight assigned by user to the response time requirement. This specifies the priority to the QoS requirement response time.
- (W_{TH} **(Weight of the Throughput):** the weight assigned by user to the throughput requirement. This specifies the priority to the QoS requirement throughput.

6.2.1 Cloud service Matchmaking Algorithm

The objective of the proposed matchmaking algorithms 1 and 2 is to generate a set of matched services from various cloud service providers through semantic matching. The matchmaker compares the input parameters given by the user with the performance parameters and pricing model of the cloud service providers to find the best matched cloud services. The proposed matchmaking algorithm expressed in Algorithm 1 was inspired by a previous study (Jrad, Tao, and Streit). Based on the limitations of the existing work, we consider the semantic matchmaking and nonfunctional attributes of the SLA in Algorithm 1, whereas the user preferences are taken into account to enhance the performance of Algorithm 2.

Matchmaking Algorithm 1

The cloud service matchmaking algorithm denoted as Algorithm 1 takes a set of cloud services, a set of cloud service providers, and a user request with QoS constraints as inputs. This algorithm generates a set of matched services as output.

As shown in Algorithm 8, for each service demanded (virtual machine or storage), the algorithm passes through all cloud service providers (lines 1 to 3) to find the providers, that semantically support the functional requirements. The algorithm checks if the QoS parameters of the providers are within the range specified by the user request, as well as the users maximum willingness to pay the cost (lines 4 to 5). The matched cloud service providers that satisfy user requirements are stored in the CloudMatchmakingList (line 6). The algorithm then returns the CloudMatchmakingList, which contains the list of providers and services that match the functional and nonfunctional requirements of the user.

Algorithm 8 QoS-based Cloud service matchmaking

Input: $sem\_S_{LIST}, sem\_P_{LIST}, R_{VM}, R_{CST}, R_{AV}, R_{TH}, R_{RT}$

Output: CloudMatchmakingList

```
1:  for each $s \in sem\_S_{LIST}$ do
2:      for each $p \in sem\_P_{LIST}$ do
3:          if $s \in p$ then
4:              if $P_{VM} \leq R_{VM} \& P_{CST} \leq R_{CST} \& P_{AV} \geq R_{AV} \& P_{RT} \leq R_{RT} \& P_{TH} \geq R_{TH}$ then
5:                  $CloudMatchmakingList.add(s, p)$;
6:              end if
7:          end if
8:      end for
9:  end for
10: if size of(CloudMatchmakingList)>0 then
11:     return CloudMatchmakingList;
12: else
13:     return Failed;
14: end if
```

The time complexity of the proposed algorithm-1 is calculated as follows. In the worst-case scenario of the first phase , for each of the n requested component services, the algorithm has to go through the supported offer list of size k for each of the l candidate providers to match the SLA requirements. This gives a worst-case complexity of O(n). The second part of the algorithm , used for selecting a random provider, has a constant time complexity of O(1). Therefore, the total worst-case complexity is: O(n). This result confirms that the time complexity of the proposed algorithm is proportional to the number of requested services.

Matchmaking Algorithm-2

In this proposed Algorithm , the user-defined threshold in the form of weights is incorporated in the user request for the specified QoS parameters as an enhancement to previous Algorithm to achieve increased user satisfaction.

For the calculation of the QoS score in Algorithm 10, the Simple Additive Weighting (SAW) method proposed in (Yoon and Hwang) is used. This technique is adopted in ((Vu, Hauswirth, and Aberer),(Ouzzani and Bouguettaya),([Zeng et al.)) to calculate the overall QoS score for local optimization. I consider the weights of QoS parameters. The weight defines the priority of the QoS requirement at the user level. If the user wants to specify a higher preference for a specific QoS parameter, then he or she assigns a higher value to the weight of that QoS parameter; the weight should be between 0 and 1. The rest of the working principle of this Algorithm is similar to previous algorithm.

The time complexity of the proposed algorithm is calculated as follow. In the worst-case scenario of the first phase, for each of the n requested component services, the algorithm has to go through the supported offer list of size k for each of the l candidate providers to match the SLA requirements. This gives a worst-case complexity of O(n). The second part of the algorithm is used for selecting a random provider, has a constant time complexity of O(1). The worst-case complexity of the proposed algorithm 8 and 2 is: O(n). This result confirms that the time complexity of the proposed algorithms-1 and 2 are proportional to the number of requested services.

Algorithm 9 User Threshold-based Cloud Service Matchmaking

Input: $sem\_S_{LIST}, sem\_P_{LIST}, R_{VM}, R_{CST}, R_{AV}, R_{TH}, R_{RT}$

Output: CloudMatchmakingList

```
1: for each $s \in sem\_S_{LIST}$ do
2:     for each $p \in sem\_P_{LIST}$ do
3:         if $s \in p$ then
4:             if $P_{VM} \leq R_{VM} \& (P_{CST} \leq R_{CST} * W_{CST}) \& (PAV \geq R_{AV} *$
   $W_{AV}) \& (P_{RT} \leq R_{RT} * W_{RT}) \& () P_{TH} \geq R_{TH} * W_{TH}$ then
5:                 $CloudMatchmakingList.add(s, p);$
6:             end if
7:         end if
8:     end for
9: end for
10: if size of (CloudMatchmakingList)$>0$ then
11:     return CloudMatchmakingList;
12: else
13:     return Failed;
14: end if
```

6.2.2 Cloud service selection

A cloud service selection problem can be described by using basic notations and algorithm as follows:

- **CloudMatchmakingList (cloud matchmaking list):** This is a list of service providers that fulfills the functional and nonfunctional requirements specified by the cloud service user. The list is obtained by using one of the two above-mentioned matchmaking algorithms.
- **Fit (customer scoring function):** This function translates the overall service quality parameter level into a corresponding satisfaction level of the customer requirements.
- **C(s) (customer willingness to pay):** This indicates the willingness of the customer to pay for an ideal service quality.
- **Cost (p) (cost of selection):** This refers to the total usage cost for candidate service selection.
- U_{value}**(utility value):** The utility value of service provider p in providing service s is calculated by using the function given in (Jrad et al.):

$$U_{value}(s,p) = C(s) \times FtotCost(p) \tag{6.1}$$

$$Ftot = W_{(AV)}(s) \times Fit_{(AV)}(s,p) + W_{(RT)} \times Fit_{(RT)}(s,p) + W_{(TH)} \times Fit_{(TH)}(s,p) \tag{6.2}$$

- **UtilityMapList:** This helps to map the utility values to the corresponding candidate service selections.
- $X_{optimal:}$ This is the optimal service selection list for the deployment of the requested service (s).

Cloud service Selection Algorithm

The objective of the cloud service selection algorithm is to select and sort the matched services based on QoS values, so as to generate an optimal selection list. In the algorithm, we perform selection using the CloudMatchmakingList produced by the semantic matchmaking algorithm. The work proposed in (Jrad et al.) had given little attention towards cloud service selection process. Only single solution is provided by previous work however, in our proposed cloud service selection approach, the algorithm selects a list of services for the cloud service composition. The proposed Algorithm 3 works with the set of selected possible service providers from the above matchmaking Algorithm 1 to obtain the optimal service selection list.

Algorithm 10 Utility-based Cloud Service selection

Input: CloudMatchmakingList, $Fit_{(AV)}, Fit_{(TH)}, Fit_{(RT)}, W_{(AV)}, W_{(TH)}, W_{(RT)}$, Cost

Output: optimal service SelectionList $X_{optimal}$

```
1:  for each $p \in CloudMatchmakingList$ do
2:      $Ftot = W_{(AV)}(s) \times Fit_{(AV)}(s,p) + W_{(RT)} \times Fi_{(RT)}(s,p) + W_{(TH)} \times Fit_{(TH)}(s,p)$;
3:      $U_{value}(s,p) = C(s) \times FtotCost(p)$;
4:      if $U_{value}$>0 then
5:          $putU_{value}inUtilityMapList(p, s, U_{value)}$;
6:      end if
7:  end for
8:  if $sizeof(UtilityMapList)$>0 then
9:      $SortUtilityMapListbyU_{value}indecreasingorder$;
10:     $X_{optimal} = UtilityMapList.firstEntry().getKey()$;
11: end if
12: return $X_{optimal}$
```

The input of the above algorithm is the customers scoring functions and their weight factors presenting the requested QoS. I calculate the utility value of all providers in the CloudMatchmakingList (lines 2 and 3); if the value is greater than zero, then the provider is added to the list, and its utility value to UtilityMapList (lines 4 to 6). The utility list set carries out mapping between the collected utility values and the corresponding service selection s. In the latter, the algorithm sorts only the positive utility values in the Utility

List in decreasing order (lines 8 to 10) in the form of key to represent the optimal service selection list.

The time complexity for a service selection request is consist of n component services and with one candidate provider, there are x = ln possible combinations to be evaluated. The time complexity is O(n). In the worst-case scenario, if all combinations are feasible, the resulted total time complexity will be O(n). The algorithm performs a merge-sort on the Utility List values (line 19 to 23). The worst-case complexity of this step is O(x: log x) and total worst-case complexity of the proposed algorithm is a result of the sum of its three steps which will be O(n:ln).

6.2.3 Cloud service Composition

In addition to previously defined terms, notations regarding the composition problem are defined below. Based on these, the composition algorithm is subsequently presented.

- **Req(Request):** the user request that contains the QoS constraints.
- X_{LIST} **(Composite service):** a set of possible composite services.
- X_{TH} **(Average Threshold):** the average threshold of the composite service.
- X_{AV} **(Average Availability):** the average availability of the composite service.
- X_{RT} **(Average Response Time):** the average response time of the composite service.

Cloud service Composition Algorithm

The objective of the cloud service composition algorithm is to compose providers of cloud services so as to generate a composition plan as a composite service. In the previous work, composition is based on the QoS requirements and cost but in the proposed, we consider the utility values along with cost.

As shown in Algorithm 11, the process begins with a search for services from the Xoptimal list. Based on the user-specified constraints and validity checking, a merge operation is carried out, which results in the addition of the service to the XLIST (composition list) (lines 1 to 5). For different combinations of compositions, the average QoS value of

Algorithm 11 Cloud Service Composition Algorithm

Input: $X_{optimal}$, Req

Output: X

```
1: for each $s \in X_{optimal}$ do
2:     if meetconstrains(s) and isValid(s) then
3:         $X_{LIST} = merge(s, X_{LIST})$; ;
4:         compose $(X_{LIST})$;
5:         $X_{TH} = TH/N; X_{AV} = AV/N; X_{RT} = RT/N$;
6:     end if
7: end for
8: Find the X that have min. no. of the providers in a $X_{LIST}$ ;
9: return X;
```

each composition is calculated (line 5). I obtain the composed service provider list as a composite service. From that list, we find the composite service X that has fewer service providers (line 6); X is returned as the result.

6.3 Cloud Ontology

The semantic matchmaking process is done by using cloud domain ontology. For the semantic matching, I use the following two ontologies:

Service Requester Ontology: The Figure 6.2 shows the service requester ontology, in which, I present the user requirements in the form of functional (e.g., type of test centers, storage and VM instances) and nonfunctional (e.g., cost, performance, latency, response time and availability) characteristics. The functional characteristics are related to a particular service and the nonfunctional information is global for a particular composition solution. To identify each service request UID (unique identifier) is there and it is related to specific type of customer.

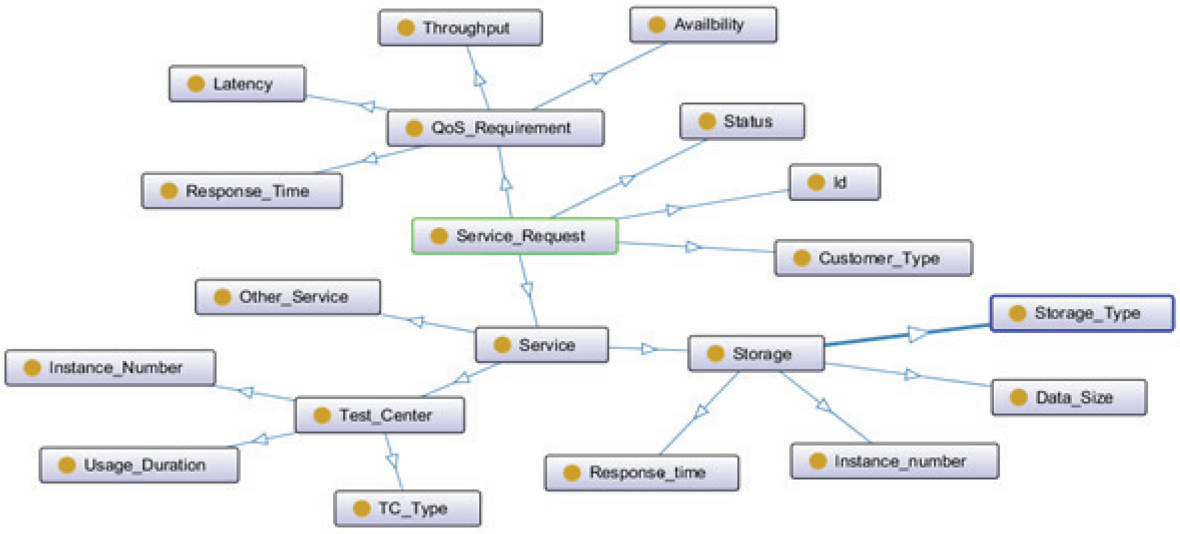


Figure 6.2: Service Requester Ontology

Cloud Service Provider Ontology: The figure 6.3 represents the cloud service provider ontology, which describes the services provided by the provider, as well as their QoS in detail. In this model, each provider is represented in the ontology by a name, ID, and location, along with their costing policies for the storage and computing resources provided by them.

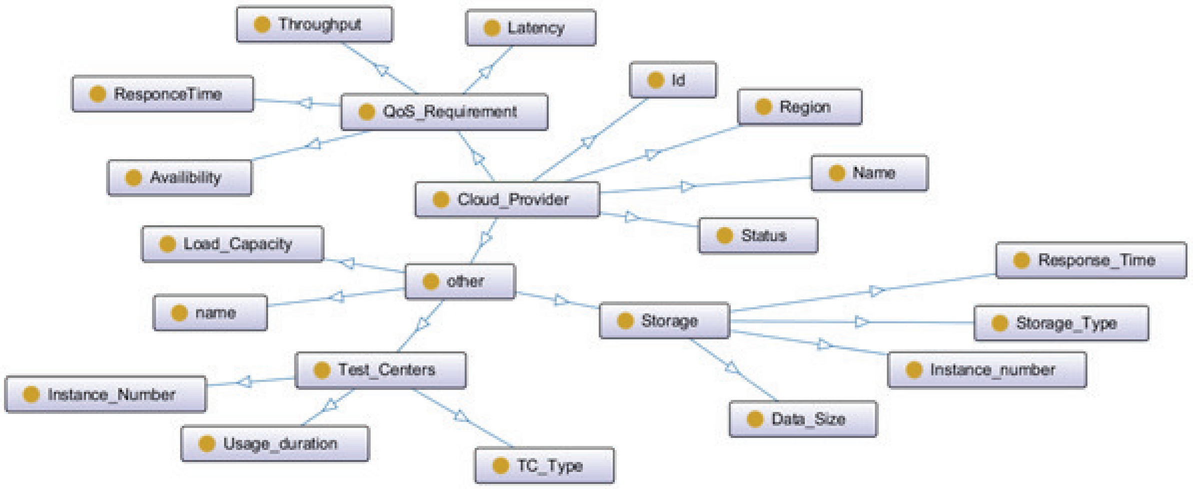


Figure 6.3: Cloud Provider Ontology

6.4 Cloud-based Healthcare Decision Making System

I considered the cloud-based healthcare decision making system in the implementation of a prototype of our proposed approach, as shown in Fig.6.4. The prototype is divided into three main parts: user, cloud broker, and infrastructure defined as follows:

- **User:** It is a computer or a person who want to use cloud services. In this cases, various types of users such as Medic, Research Scientist and Admin interact with the Cloud Broker to access services through User Interface.

- **Cloud Broker:** It is an intermediate entity between the Cloud service providers and users to facilitate functionalities such as matchmaking, selection and composition of services.

- **Semantic Matchmaking:** The semantic matchmaking is used to semantically match a set of service from the cloud service repository with users request. If the service is matched, then it added to the candidate list to generate a set of matched services.

- **Composition:** It is responsible to compose service providers. The output of this module can be a composition plan, or a reference to a composite solution.

- **Selection & Ranking:** Service selection is responsible to select and filter a set of matched services. After selection, services are ranked & filtered based on requesters QoS constraints.

- **Ontology:** The ontology act as a knowledge-base to perform semantic matchmaking and to describe semantic description of cloud services.

- **Service Provider Repository:** This repository contains the semantically annotated Cloud services.

- **VM (Test centers):** The test center facility, such as Blood-Test, X-Ray, CT-scan etc created by the cloud service providers with the help of virtual machines instantiated at the datacenter (infrastructure).

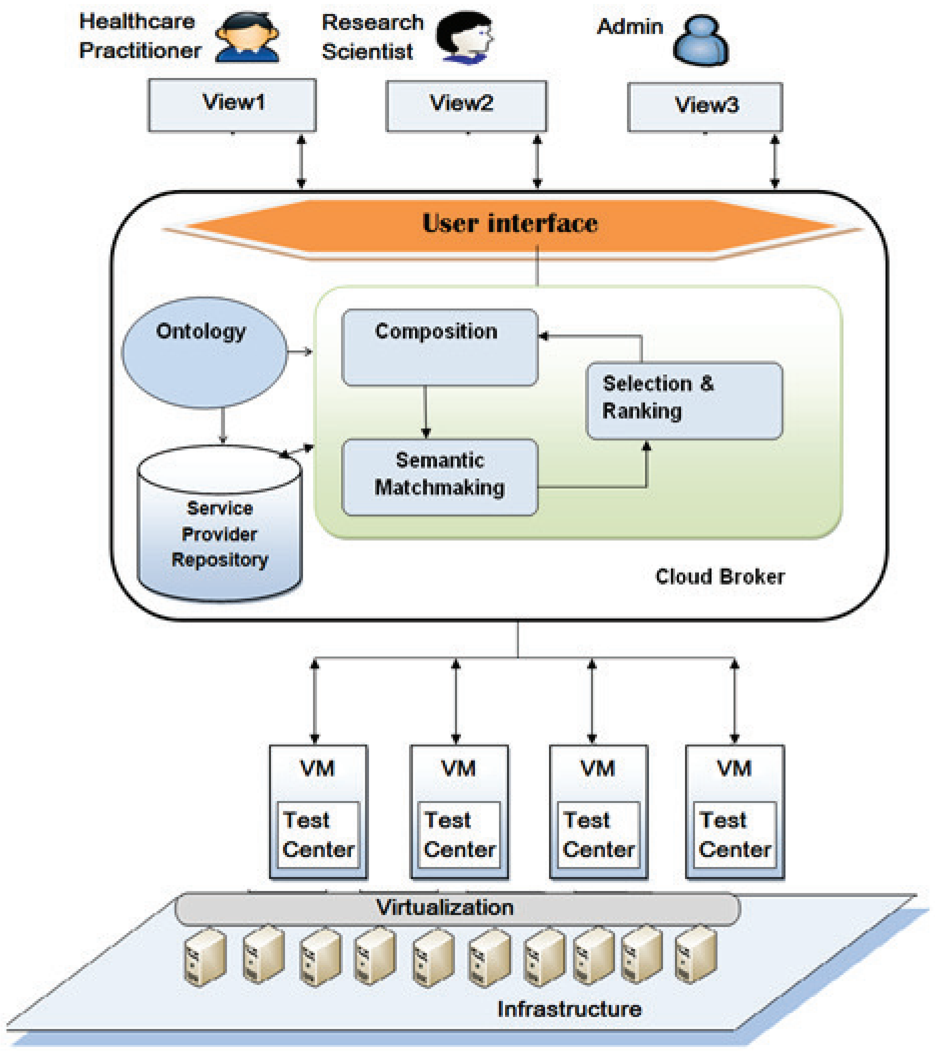


Figure 6.4: Prototype Implementation

For simplicity, three types of users are identified in our prototype: user 1 (Medic), Healthcare Practitioners; user 2, Research Scientist and user 3, Admin. As shown in Fig. 5, patients can interact with and submit a request to the broker through the user interface (UI). The cloud broker is equipped with service matchmaking, selection, and composition modules, along with a domain ontology and service provider repository. In the proposed system, the ontology works as a knowledge-base to provide semantic descriptions for broker-level functionalities. The test center facility is delivered by the cloud service providers with the help of virtual machines created at the datacenter. The Healthcare Decision Making System (HDMS) is suitable for regions, in which there are no diagnostic facilities. For a better understanding, let us take as an example a regional area with a primary dispensary, from which local people are able to obtain primary treatment, but without such facilities as a test center, research laboratory, and blood bank. For diagnosis and treatment of severe conditions, patients can provide samples for basic tests, such as blood and urine analysis, as well as obtain a recommendation for further tests from a medic. The medic may take the samples for analysis or recommend more advanced tests, such as ECG or MRI, and may provide the patient details to a more distant health facility for diagnosis. With a cloud-based decision-making system, however, the medic can submit the symptoms presented by the patient and then generate a diagnostic report for appropriate treatment.

In this system, the user has to provide the patient profile and symptoms. According to the user input, the system finds information on the possible disease of the patient, the doctor consultants available, the tests that need to be done, and the estimated cost of treatment.

6.4.1 Experimental Work and Results

The CloudSim toolkit (Calheiros et al.) is used to implement the proposed approach. The core services provided by the cloudsim are the SLA manager, deployment manager and the match maker implemented as Java classes within the cloud service broker package.

Experimental Setup

The various experiments are carried out on a machine equipped with various hardware and software, such as Eclipse Juno, JDK 1.7, CloudSim 3.0 (with the OCCI package), and an Intel Core i7 2.40-GHz processor with 8 GB RAM.

By using Java, I created an environment suitable for a multi-cloud environment framework. This allowed to validate the functionality of the components without having to setup a real cloud, which is time-consuming and costly. Moreover, simulation environment supports to validate the proposed broker model in different scenarios.

Performance Parameters

Following performance parameters have been considered to measure the performance of the approach.

(i) **Availability :** It defines the accessible services over the time in percentage. Higher value of availability increases the performance.

(ii) **Response Time:** It defines the time required to perform the process. i.e. Discovery, Selection and Composition.

(iii) **Throughput:** For service users, a throughput is defined as the volume of work that a service can perform in a given period of time. Higher throughput shows higher performance of the process.

6.5 Prototype Implementation of the Healthcare Decision Making System

In the HDMS system, the user has to submit the patient name and at least two symptoms presented. The processing engine carries out the identification of the disease, after which the system performs matchmaking between the user request and the available test centers. The processing engine discovers the possible disease and also suggests doctors for treatment based on the available record. The diagnostic report for the patient includes the required tests and the estimated cost of treatment. Thus, Healthcare Decision-Making System plays a vital role in the healthcare sector and provides a cost-effective solution for patients.

6.6 Cloudsim simulation

The cloudsim is a Java-based tool for modeling and simulation of large-scale cloud computing infrastructures on a single host. In addition, the support for custom developed scheduling and allocation policies in the simulation made cloudsim an attractive tool for cloud researchers. In the simulation environment, cloudsim is used to model large-scale and heterogeneous cloud providers.

(i.) **Cloudsim configuration** In this experiment, test centers, client, and service providers are defined as follows:

(a.) **VM (Test Centers):** Different types of VMs created over the host is shown in Table 6.1, which represents various test centers.

Table 6.1: Test Centers detail

TC No.	TC_ Name	TC_ MIPS	TC_ PES	TC_ UNIT	TC_ RAM	TC_ BW	TC_ SIZE	TC_ OS	TC_ ARCH
1	BloodTest	0.5F	1	1	0.5F	100000	25F	Linux	x64
2	X-Ray	1F	1	1	0.5F	100000	25F	Win	x64
3	CitiScan	1F	1	2	1F	100000	50F	Lnux	x64
4	MRI	1F	1	2	1F	100000	50F	Win	x64
5	Sonography	1F	1	3	1.7F	100000	75F	Linux	x64
6	DNA	1F	1	3	1.7F	100000	75F	Win	x64
7	Neurology	1F	2	4	2F	100000	100F	Linux	x64
8	Kidney	1F	2	4	2F	100000	100F	Win	x64
9	Bone	1F	2	6	3.75F	100000	150F	Linux	x64
10	Cancer	1F	2	6	3.75F	100000	150F	Win	x64
11	Eye	1F	4	8	4F	100000	200F	Linux	x64
12	Dental	1F	4	8	4F	100000	200F	Win	x64
13	Gynac	1F	4	12	7.5F	100000	250F	Linux	x64
14	ECG	1F	4	12	7.5F	100000	250F	Win	x64
15	TC1	1F	8	16	8F	100000	300F	Linux	x64
16	TC2	1F	8	16	8F	100000	300F	Win	x64
17	TC3	1F	8	20	15F	100000	350F	Linux	x64
18	TC4	1F	8	20	15F	100000	350F	Win	x64

Virtual machine is a software computer that likes a physical computer. VM runs operating system have different architecture, size, bandwidth and ram. In CloudSim to configure each VM (Test Center), we have to specify the TC_Name (Test Center Name), TC_MIPS (Machine Instruction per sec), TC_PE (number of processing elements), TC_RAM (RAM of Test center), TC_BW (Bandwidth of Test center), TC_SIZE(Test Center Size), TC_OS (Operating system of test center), TC_ARCH (architectures of testcenter). The number of VMs can be increased to fulfill the user requests.

(b.) **Client:**

Various cloud service users are defined in Table 6.2. In this table, clients weight of availability (Client_WAV), response time (Client_WRT), throughput (Client_WTH) is also specified. Client is end user who uses the functionality that provided by cloud service providers. Different clients have various QoS requirements.

Table 6.2: Client Requirement

No.	**Client Name**	**AV**	**RT**	**TH**	**Willing ness**	**Region**	**Client_ WAV**	**Client_ WRT**	**Client_ WTH**
1	**Doctor**	**96**	**3.5**	**16**	**0.06**	**2**	**0.33**	**0.33**	**0.33**
2	**Pharmasist**	**94**	**25**	**8.2**	**0.03**	**2**	**0.4**	**0.2**	**0.4**
3	**Receptionist**	**91**	**10**	**1.8**	**0.025**	**2**	**0.2**	**0.6**	**0.2**
4	**Admin**	**80**	**50**	**10**	**0.07**	**3**	**0.33**	**0.33**	**0.33**
5	**Pethologist**	**85**	**55**	**5**	**0.03**	**0**	**0.4**	**0.4**	**0.4**

(c.) **Cloud provider:**

The state detail of cloud providers is shown in Table 6.3. Different cloud service providers have different operating systems, architectures, cores and QoS parameters are retrieved from (Jrad et al.).

Table 6.3: Cloud service provider Detail

Provider Name	ID	Region	OS	Arch	U Cores	F Cores	Max CPU	AV	RT	TH
AMAZON_EU	0	2	Lin-Win	x64	2	598	1860	99.97	3.63	19.11
AMAZON_US	1000	0	Lin-Win	x64	0	600	1860	99.98	8.59	2.39
AMAZON_JP	2000	3	Lin-Win	x64	0	600	1860	99.91	21.19	2.73
ELASTIC_EU	3000	2	Lin-Win	x64	0	600	1860	99.99	2.87	15.42
ELASTIC_US	4000	0	Lin-Win	x64	0	600	1860	99.97	12.12	1.41
GOGRID_EU	5000	2	Lin-Win	x64	0	600	1860	99.96	1.45	48.52
GOGRID_US	6000	0	Lin-Win	x64	0	600	1860	100	8.35	3.13
RACKSPA_EU	7000	2	Lin-Win	x64	0	600	1860	99.97	2.73	11.28
RACKSPA_US	8000	0	Lin-Win	x64	0	600	1860	99.96	11.32	1.76
CSIGMA_EU	9000	2	Lin-Win	x64	2	598	1860	99.95	2.69	29.2
CSIGMA_US	10000	0	Lin-Win	x64	0	600	1860	99.93	12.58	1.58
VOXCLO_EU	11000	2	Linux	x64	0	600	1860	99.93	4.76	36.05
VOXCLO_US	12000	0	Linux	x64	0	600	1860	99.93	8.27	1.96
VOXCLO_SG	13000	3	Linux	x64	0	600	1860	99.8	24.65	0.63
OPSOURC_EU	14000	2	Lin-Win	x64	0	600	1860	99.98	2.91	27.5
OPSOURC_US	15000	0	Lin-Win	x64	0	600	1860	99.96	8.64	2.47
OPSOURC_AU	16000	5	Lin-Win	x64	0	600	1860	99.95	28.89	0.85
CITYCLO_EU	17000	2	Lin-Win	x64	4	596	1860	99.93	4.18	23.13

HPCLOUD_EU	**18000**	**0**	**Linux**	**x64**	**4**	**596**	**1860**	**100**	**9.08**	**1.34**
FLEXISCA_EU	**19000**	**2**	**Lin-Win**	**x64**	**4**	**596**	**1860**	**99.5**	**4.04**	**24.51**

6.6.1 Experimental Results

This section focuses on detail about dataset and various steps to be performed to show the applicability of the proposed work.

Table 6.4: Performance evaluation of proposed prototype

User Input	**Diagnosis Result**	**Tests Required**	**Estimated Cost of Treatment (Rs)**
Cold, Fever	**Viral infection**	**Blood Test, Urine Test**	**400**
Cold, Fever	**Typhoid**	**Blood Test, Urine Test, Hemoglobin**	**1500**
Bleeding, Tooth pain	**Root canal**	**X-ray, Blood pressure**	**2500**
Stomach pain, Loose motion	**Food poison**	**Sonography, Blood test**	**1000**
Chest pain, Breathing problem	**Heart Attack**	**ECG,CT-scan,MRI, Blood pressure**	**50000**

The Table 6.4 shows the performance of the various processes, i.e. matchmaking, selection and composition of the prototype. Based on the user inputs and the user specified constraints of availability (AV), response time (RT) and throughput (TH), selection, composition solution will be generated. In the proposed work, I have taken 18 service providers for the test centers. Results show the diagnosis of the patient disease, Test required for the treatment, estimate treatment cost and number of possible test centers for the treatment taken by various processes. Finally, the status of patient would be generated.

6.6.2 Performance evaluation of Proposed Approaches

This section shows the performance details of proposed algorithms for the various number of requests along with the Average values of non functional parameters like cost/hour, availability, response time in second, throughput in Mbit/s.

Table 6.5: Proposed Algorithm-1 Results

Parameter	**Values**	**Values**	**Values**
Number of Requests	**10**	**100**	**1000**
No. of deployed Requests	**10**	**96**	**864**
No. of failed Requesst	**0**	**0**	**3**
No. of matched Requests	**10**	**96**	**867**
No. of matched Providers	**5**	**11**	**12**
Average matched cost	**0.117**	**0.233**	**0.235**
Average matched Availability	**99.82%**	**99.90%**	**99.92%**
Average matched Response Time	**3.09**	**3.26**	**3.65**
Average matched Throughput	**28.55**	**27.88**	**26.01**

The Table 6.5 represents the performance evolution of proposed matchmaking algorithm-1. I have taken values of of request from 10 to 1000. For all these requests, I have generated values for the number of deployed request, failed request, and matched request. For each request, values of QoS parameters, i.e. cost, availability, response time, throughput are calculated ans average of each is considered in the result.

As shown in Table 6.5, I have proposed the approach for matchmaking of cloud service providers based on user request where number of providers have been matched based on the total number of deployed request over the cloud. As a result, I have retrieved number of matched requests along with given cost, availability, response time and throughput.

Table 6.6: Proposed Algorithm-2 Results

Parameter	**Values**	**Values**	**Values**
Number of Requests	**10**	**100**	**1000**
No. of deployed Requests	**10**	**96**	**864**
Number of failed Requests	**0**	**0**	**3**
No. of matched Requests	**10**	**96**	**867**
No. of matched Providers	**6**	**11**	**12**
Average matched cost	**0.115**	**0.240**	**0.251**
Avg. matched Availability	**99.92%**	**99.91%**	**99.90%**
Avg. matched Response Time	**2.66**	**3.36**	**3.68**
Avg. matched Throughput	**32.20**	**27.90**	**26.56**

The table 6.6 represents the performance evaluation of proposed matchmaking algorithm-2. I have taken values of request from 10 to 1000. For all these requests, we have generated values for the number of deployed request, failed request, and matched request. For each request, values of QoS parameters, i.e. cost, availability, response time, throughput are calculated ans average of each is considered in the result. The table 6.6 represents the performance evaluation of proposed matchmaking approach 2. For the request values of 10,100 and 1000, we generated the number of deployed request, failed request, and matched request. All the request are served by matched providers.

For each request, cost, availability, response time, throughput are calculated average of each is considered in the result.

6.6.3 Comparison of Matchmaking approaches with an existing approach

The comparative analysis among the proposed work and existing work related to Cloud service matchmaking is presented in Figure. Figure 6.5 presents a comparison between the

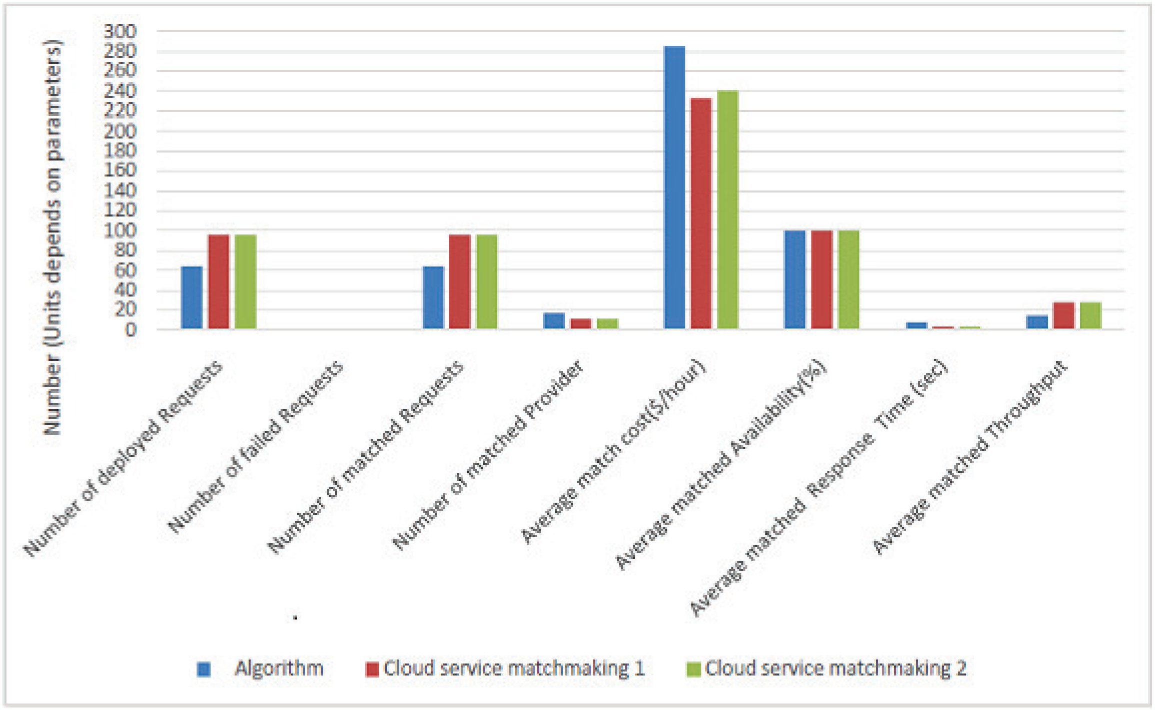


Figure 6.5: Comparison Results

existing algorithms and the proposed matchmaking algorithms for 100 requests. The graph indicate that the proposed algorithms performs better than the existing Algorithm (Jrad, Tao, and Streit),(Jrad et al.).

Table 6.7: Cloud service selection Results

Parameter	**Values**	**Values**	**Values**
Number of Requests	**10**	**100**	**1000**
Number of deployed Requests	**10**	**96**	**804**
Number of failed Requests	**0**	**0**	**0**

Number of selected Requests	**10**	**96**	**804**
Number of selected Providers	**4**	**4**	**7**
Average selected cost	**0.081**	**0.174**	**0.170**
Average selected Availability	**99.90%**	**99.88%**	**99.88%**
Average selected Response Time	**3.91**	**3.85**	**4.02**
Average selected Throughput	**23.07**	**23.81**	**22.51**

As shown in Table 6.7, I have proposed an approach to select the cloud service providers by taking the result received from the matchmaking process, where number of providers have been selected based on the total number of deployed request over the cloud.

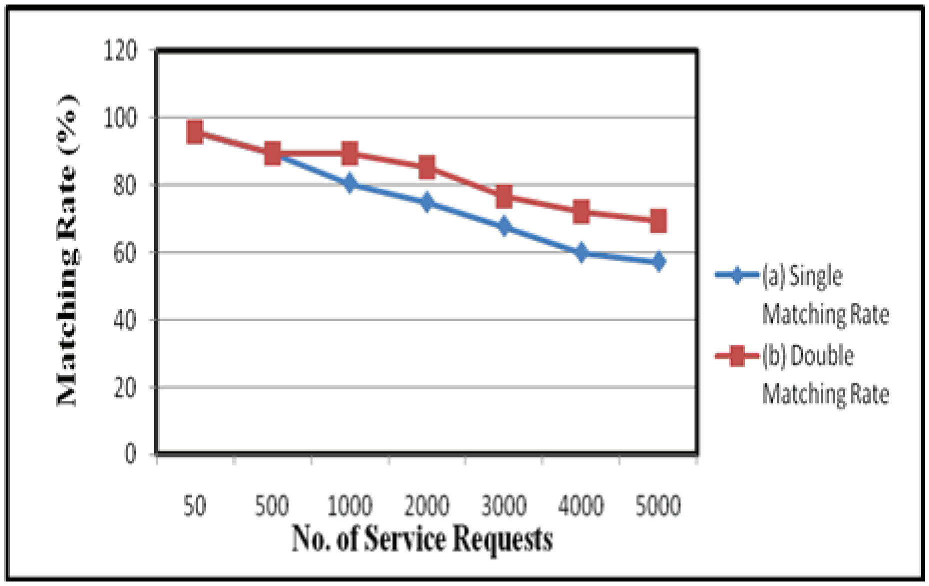


Figure 6.6: Matching Rate of Selection Algorithm

The matching rate ,which defines the percentage of matched requests,goes down with the increasing of number of requests for all cases as depicted in figure 6.6. The results show that the matching rate is higher for Double case that indicates the it perform better with customer requests when customers have a larger willings to pay. The Figure 6.7 shows the number of different providers selected by each algorithm with different number of service requests. For the Single case , it only selects upto 12 providers but for Double Case , it

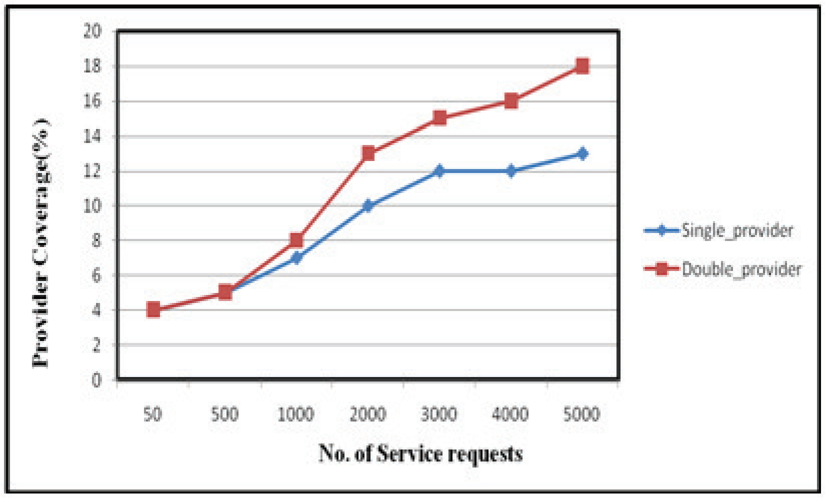


Figure 6.7: Provider Coverage of Selection Algorithm

raises upto 18 providers which gives better result. This result shows that more number of providers will be selected when the number of requests is large.

Table 6.8: Proposed Algorithm-4 Results

Parameter	Values	Values	Values
Number of Requests	10	100	1000
Number of deployed Requests	10	100	864
Number of failed Requests	0	0	3
Number of composed Requests	9	96	867
Number of composed Providers	7	11	52
Average composed cost	0.187	0.714	3.991
Average composed Availability	99.96%	99.96%	99.89%
Average composed Response Time	1.44	2.46	3.42
Average composed Throughput	48.51	30.61	26.17

The Table 6.8 shows the performance of composition algorithm. For the request values 10,100 and 1000, I have generated the number of deployed request, failed request, and matched request. The cost, availability, response time and throughput are calculated for each request and average of that is considered.

As shown in Table 6.8, I have proposed the approach for composition of cloud service providers based on outcome of selection process, where number of providers has been composed based on the total number of deployed request over the cloud. As a result, I have retrieved the number of composed requests along with given cost, availability, response time and throughput.

Workflow plan of cloud service based Healthcare System The Figure 6.8 shows the data-flow graph for composition plan. Directed acyclic graph (DAGs) id represents data-flow graph. The start node denotes the source service; the node where the edge ends is denoted as destination service. The data items transferred between these abstract application services form a set D = data i ; 1 <= i <= d.

A service can be executed on multiple virtual machines and databases. After mapping each service, VM and Database need to be selected for each service. As a result, any solution to a composition problem includes: Schedule the execution order of the services. This execution order is the data-flow graph.

(i) **SS(Schedule String):**

SS represents the schedule string of the solution, e.g., the execution order of this solution in Fig. 9 is P0; P1000; P50000; P9000; P12000; P14000; P17000 ; P19000.

(ii) **MS (Mapping String):**

$P0 \rightarrow vm1, \qquad db1P1000 \rightarrow vm2, db1$

$P50000 \rightarrow vm3, db1P9000 \rightarrow vm2, db1$

$P12000 \rightarrow vm3, db1P14000 \rightarrow vm1, db1$

$P17000 \rightarrow vm1, db1P19000 \rightarrow vm1, db1$

In this solution, MS represents the mapping string, e.g., provider P0 is mapped and further deployed on virtual machine vm1 and database db1. Same procedure is done for the other

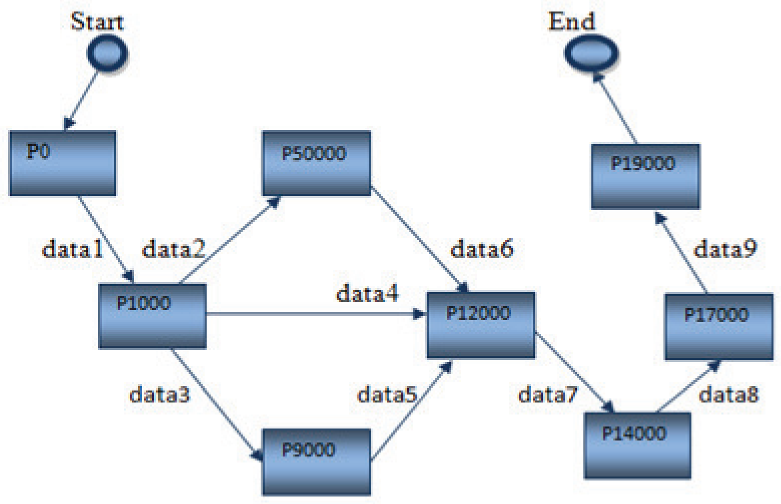


Figure 6.8: Cloud Composition plan

providers of service. Vm1 represents basic test service, vm2 represents critical test service and vm3 represents advanced test service.

6.6.4 Comparison with existing work

The comparative analysis among the proposed work and existing work is summarized in Table 6.9. A comparison has been presented in Table 6.9 among the existing algorithm and proposed algorithms, which show that the proposed work performs better than exiting one. For the 10 requests, the execution time of matchmaking algorithm-1 is less. As long as increasing number of requests execution time of both proposed algorithm is nearly same. For the execution time, the proposed algorithm perform faster than the existing solution.

Table 6.9: Comparison Results

Parameter	**Existing Algo. (Jrad, Tao, and Streit)**	**Proposed Algorithm-1**	**Proposed Algorithm-1**
Number of Requests	**100**	**100**	**100**
Number of deployed Requests	**64**	**96**	**96**
Number of failed Requests	**0**	**0**	**0**
Number of matched Requests	**64**	**96**	**96**
Number of matched Providers	**17**	**11**	**11**
Average matched cost	**0.285$/hour**	**0.233$/hour**	**0.240$/hour**
Average matched Availability	**99.9%**	**99.9%**	**99.9%**
Avg. matched Response Time	**8.00**	**3.26**	**3.36**
Average matched Throughput	**14.40**	**27.88**	**27.90**

6.7 Discussions

In this chapter, I have presented a framework for service discovery, service selection and service composition for Cloud services by considering Semantic Web and QoS-parameters, such as response time and throughput for service selection, discovery and composition. I have proposed the approaches to realize the framework. The approach has been evaluated by developing the prototypes for Healthcare Decision Making System using CloudSim environment to generate the multiple cloud environment, where cloud services are provided by multiple cloud service providers. I have compared the proposed system with existing system to demonstrate the novelty and need of the system. As per our knowledge, this is the novel concept proposed by me to present the approach for Cloud services by considering service discovery, selection and composition processes in integrated way.

Chapter 7

Summary and Conclusions

In this chapter, I present the summary and conclusions related to the Service discovery, selection and composition of Heterogeneous Web services. Section 7.1 discusses the summery of Heterogeneous Web services discovery, selection and composition; Section 7.2 presents conclusions and future work of the proposed work.

7.1 Summary

Web services play a key role in the emerging technologies, such as Cloud computing and Internet of Things (IOT). Over the time, researchers have proposed approaches and solutions related to Web service discovery, service selection and service composition problems by considering them as key challenges of Web service research. I have discussed my relevant contribution in the related work section. Literature related to Web services, such as SOAP-based Web services, RESTful Web services and emerging Web services on Cloud (Cloud services) as Heterogeneous Web services are discussed. Existing work has not covered key research challenges collectively for the Heterogeneous services.I have presented the contribution in the form of proposed framework and approaches related to Web service discovery, service selection and service composition by incorporating Semantic Web and non-functional characteristics. For the performance measurement and demonstration, standard as well as open data sets have been used in the experimental work. Based on the proposed work, prototypes related to Public domain and Healthcare domain have been developed with appropriate comparison with relevant work. Analysis of the achieved re-

sults shows that proposed work performs well in demonstrated environment. Concluding remarks and future directions are described in the following section.

7.2 Conclusions and Future Work

This sections presents the conclusions derived from the proposed work and future directions for further investigation.

7.2.1 SOAP-based Web services

As a proposed work, I have presented an integrated approach on the support of runtime Web service discovery, selection and composition based on Semantic Web and non-functional characteristics to facilitate the end user to search, select and compose the services with increased satisfaction. We have proposed a framework to show the inter-relationship among the discovery, selection and composition tasks.Various Most of the frameworks proposed for the service discovery, selection and composition have considered these tasks on individual basis. The proposed framework has considered these service tasks collectively. Based on the proposed framework, I have presented the approaches for service discovery, selection and composition by incorporating ontology as knowledge base and non-functional characteristics.I have developed a prototype for the Healthcare Information System to offer the healthcare services to end user for performing the routing task such as to make the appointment and treatment from the nearest Healthcare centers. I have performed the experiments on the proposed approach using standard datasets such as WSC2009 to evaluate the performance and comparison with existing work. We conclude that runtime service discovery, selection and composition using non-functional characteristics can be achieved. However, with this work I demonstrated that end user has to play an important role to generate the efficient composition solution. I have evaluated the proposed work using real life scenario to show the evidence of its usability. I have made comparison of proposed approach with existing approaches to demonstrate the effectiveness of the proposed work.

As a future work, it could be possible to extend the proposed system through integration of hand-held devices to get the advantage of pervasiveness. Cloud-based platform could also be incorporated to provide services to large scale level from local access to

remote area where healthcare facility is not easily accessible.

7.2.2 RESTful Web services

I have presented the efficient and effective approaches for automated service discovery, selection and composition using Linked Open Data and RESTful Web services as well as to describe services in the form of RDF data to link them with Linked Data Cloud. As a part of the proposed work, various data sources are transformed into RDF triple store and interlinked them into the LOD cloud. RDF provides uniform data model which describes data semantically as well as avoids the issue of data heterogeneity. To demonstrate the performance of the proposed work, a prototype for Population Information System using US Census dataset is developed which allows end user to retrieve population related information from the publicly available data by providing the input parameters. A prototype for Healthcare Recommendation System based on Linked Open Data Cloud is also implemented which facilitate end user to get recommendation for epidemiology i.e. H1N1 as per the user query. I have used proposed approach for two different kinds of datasets. From these two prototypes, I have analysed that proposed approach is more beneficial for Healthcare sector, where limited datasets and features are restricting end user to access the resources effectively. The results achieved through experimental work and prototype development demonstrates the better performance of the proposed work which is also compared with the existing approaches concerned with Linked Data based RESTful Services. In this way, an attempt has been made by me to integrate and automate the discovery, selection and composition tasks of the RESTful services using Linked Open Data.

As a future work, I can extend the proposed work by incorporating emerging concepts like Internet of Things, which use RDF data and REST principles to make it more effective and beneficial to the end users work.

7.2.3 Cloud Services

A framework and approaches for the cloud service discovery, selection and composition by considering the QoS parameters of SLA, user preferences and domain ontology have been proposed. Experimental results show that the proposed work performs well in comparison

with existing work for the matchmaking process. A prototype for Decision Making Healthcare System has been developed by using the proposed work, where I have considered the basic health centre scenario to facilitate the end user for the diagnose and decision making facility.

As a future work, nonlinear evolutionary techniques could be incorporated for optimization requirement. Several cloud related issues, i.e. scalability, interoperability can be considered to improve the proposed approach.

Chapter 7

Summary and Conclusions

In this chapter, I present the summary and conclusions related to the Service discovery, selection and composition of Heterogeneous Web services. Section 7.1 discusses the summery of Heterogeneous Web services discovery, selection and composition; Section 7.2 presents conclusions and future work of the proposed work.

7.1 Summary

Web services play a key role in the emerging technologies, such as Cloud computing and Internet of Things (IOT). Over the time, researchers have proposed approaches and solutions related to Web service discovery, service selection and service composition problems by considering them as key challenges of Web service research. I have discussed my relevant contribution in the related work section. Literature related to Web services, such as SOAP-based Web services, RESTful Web services and emerging Web services on Cloud (Cloud services) as Heterogeneous Web services are discussed. Existing work has not covered key research challenges collectively for the Heterogeneous services.I have presented the contribution in the form of proposed framework and approaches related to Web service discovery, service selection and service composition by incorporating Semantic Web and non-functional characteristics. For the performance measurement and demonstration, standard as well as open data sets have been used in the experimental work. Based on the proposed work, prototypes related to Public domain and Healthcare domain have been developed with appropriate comparison with relevant work. Analysis of the achieved re-

sults shows that proposed work performs well in demonstrated environment. Concluding remarks and future directions are described in the following section.

7.2 Conclusions and Future Work

This sections presents the conclusions derived from the proposed work and future directions for further investigation.

7.2.1 SOAP-based Web services

As a proposed work, I have presented an integrated approach on the support of runtime Web service discovery, selection and composition based on Semantic Web and non-functional characteristics to facilitate the end user to search, select and compose the services with increased satisfaction. We have proposed a framework to show the inter-relationship among the discovery, selection and composition tasks.Various Most of the frameworks proposed for the service discovery, selection and composition have considered these tasks on individual basis. The proposed framework has considered these service tasks collectively. Based on the proposed framework, I have presented the approaches for service discovery, selection and composition by incorporating ontology as knowledge base and non-functional characteristics.I have developed a prototype for the Healthcare Information System to offer the healthcare services to end user for performing the routing task such as to make the appointment and treatment from the nearest Healthcare centers. I have performed the experiments on the proposed approach using standard datasets such as WSC2009 to evaluate the performance and comparison with existing work. We conclude that runtime service discovery, selection and composition using non-functional characteristics can be achieved. However, with this work I demonstrated that end user has to play an important role to generate the efficient composition solution. I have evaluated the proposed work using real life scenario to show the evidence of its usability. I have made comparison of proposed approach with existing approaches to demonstrate the effectiveness of the proposed work.

As a future work, it could be possible to extend the proposed system through integration of hand-held devices to get the advantage of pervasiveness. Cloud-based platform could also be incorporated to provide services to large scale level from local access to

remote area where healthcare facility is not easily accessible.

7.2.2 RESTful Web services

I have presented the efficient and effective approaches for automated service discovery, selection and composition using Linked Open Data and RESTful Web services as well as to describe services in the form of RDF data to link them with Linked Data Cloud. As a part of the proposed work, various data sources are transformed into RDF triple store and interlinked them into the LOD cloud. RDF provides uniform data model which describes data semantically as well as avoids the issue of data heterogeneity. To demonstrate the performance of the proposed work, a prototype for Population Information System using US Census dataset is developed which allows end user to retrieve population related information from the publicly available data by providing the input parameters. A prototype for Healthcare Recommendation System based on Linked Open Data Cloud is also implemented which facilitate end user to get recommendation for epidemiology i.e. H1N1 as per the user query. I have used proposed approach for two different kinds of datasets. From these two prototypes, I have analysed that proposed approach is more beneficial for Healthcare sector, where limited datasets and features are restricting end user to access the resources effectively. The results achieved through experimental work and prototype development demonstrates the better performance of the proposed work which is also compared with the existing approaches concerned with Linked Data based RESTful Services. In this way, an attempt has been made by me to integrate and automate the discovery, selection and composition tasks of the RESTful services using Linked Open Data.

As a future work, I can extend the proposed work by incorporating emerging concepts like Internet of Things, which use RDF data and REST principles to make it more effective and beneficial to the end users work.

7.2.3 Cloud Services

A framework and approaches for the cloud service discovery, selection and composition by considering the QoS parameters of SLA, user preferences and domain ontology have been proposed. Experimental results show that the proposed work performs well in comparison

with existing work for the matchmaking process. A prototype for Decision Making Healthcare System has been developed by using the proposed work, where I have considered the basic health centre scenario to facilitate the end user for the diagnose and decision making facility.

As a future work, nonlinear evolutionary techniques could be incorporated for optimization requirement. Several cloud related issues, i.e. scalability, interoperability can be considered to improve the proposed approach.

Chapter 7

Summary and Conclusions

In this chapter, I present the summary and conclusions related to the Service discovery, selection and composition of Heterogeneous Web services. Section 7.1 discusses the summery of Heterogeneous Web services discovery, selection and composition; Section 7.2 presents conclusions and future work of the proposed work.

7.1 Summary

Web services play a key role in the emerging technologies, such as Cloud computing and Internet of Things (IOT). Over the time, researchers have proposed approaches and solutions related to Web service discovery, service selection and service composition problems by considering them as key challenges of Web service research. I have discussed my relevant contribution in the related work section. Literature related to Web services, such as SOAP-based Web services, RESTful Web services and emerging Web services on Cloud (Cloud services) as Heterogeneous Web services are discussed. Existing work has not covered key research challenges collectively for the Heterogeneous services.I have presented the contribution in the form of proposed framework and approaches related to Web service discovery, service selection and service composition by incorporating Semantic Web and non-functional characteristics. For the performance measurement and demonstration, standard as well as open data sets have been used in the experimental work. Based on the proposed work, prototypes related to Public domain and Healthcare domain have been developed with appropriate comparison with relevant work. Analysis of the achieved re-

sults shows that proposed work performs well in demonstrated environment. Concluding remarks and future directions are described in the following section.

7.2 Conclusions and Future Work

This sections presents the conclusions derived from the proposed work and future directions for further investigation.

7.2.1 SOAP-based Web services

As a proposed work, I have presented an integrated approach on the support of runtime Web service discovery, selection and composition based on Semantic Web and non-functional characteristics to facilitate the end user to search, select and compose the services with increased satisfaction. We have proposed a framework to show the inter-relationship among the discovery, selection and composition tasks.Various Most of the frameworks proposed for the service discovery, selection and composition have considered these tasks on individual basis. The proposed framework has considered these service tasks collectively. Based on the proposed framework, I have presented the approaches for service discovery, selection and composition by incorporating ontology as knowledge base and non-functional characteristics.I have developed a prototype for the Healthcare Information System to offer the healthcare services to end user for performing the routing task such as to make the appointment and treatment from the nearest Healthcare centers. I have performed the experiments on the proposed approach using standard datasets such as WSC2009 to evaluate the performance and comparison with existing work. We conclude that runtime service discovery, selection and composition using non-functional characteristics can be achieved. However, with this work I demonstrated that end user has to play an important role to generate the efficient composition solution. I have evaluated the proposed work using real life scenario to show the evidence of its usability. I have made comparison of proposed approach with existing approaches to demonstrate the effectiveness of the proposed work.

As a future work, it could be possible to extend the proposed system through integration of hand-held devices to get the advantage of pervasiveness. Cloud-based platform could also be incorporated to provide services to large scale level from local access to

remote area where healthcare facility is not easily accessible.

7.2.2 RESTful Web services

I have presented the efficient and effective approaches for automated service discovery, selection and composition using Linked Open Data and RESTful Web services as well as to describe services in the form of RDF data to link them with Linked Data Cloud. As a part of the proposed work, various data sources are transformed into RDF triple store and interlinked them into the LOD cloud. RDF provides uniform data model which describes data semantically as well as avoids the issue of data heterogeneity. To demonstrate the performance of the proposed work, a prototype for Population Information System using US Census dataset is developed which allows end user to retrieve population related information from the publicly available data by providing the input parameters. A prototype for Healthcare Recommendation System based on Linked Open Data Cloud is also implemented which facilitate end user to get recommendation for epidemiology i.e. H1N1 as per the user query. I have used proposed approach for two different kinds of datasets. From these two prototypes, I have analysed that proposed approach is more beneficial for Healthcare sector, where limited datasets and features are restricting end user to access the resources effectively. The results achieved through experimental work and prototype development demonstrates the better performance of the proposed work which is also compared with the existing approaches concerned with Linked Data based RESTful Services. In this way, an attempt has been made by me to integrate and automate the discovery, selection and composition tasks of the RESTful services using Linked Open Data.

As a future work, I can extend the proposed work by incorporating emerging concepts like Internet of Things, which use RDF data and REST principles to make it more effective and beneficial to the end users work.

7.2.3 Cloud Services

A framework and approaches for the cloud service discovery, selection and composition by considering the QoS parameters of SLA, user preferences and domain ontology have been proposed. Experimental results show that the proposed work performs well in comparison

with existing work for the matchmaking process. A prototype for Decision Making Healthcare System has been developed by using the proposed work, where I have considered the basic health centre scenario to facilitate the end user for the diagnose and decision making facility.

As a future work, nonlinear evolutionary techniques could be incorporated for optimization requirement. Several cloud related issues, i.e. scalability, interoperability can be considered to improve the proposed approach.

Bibliography

Adamczyk, Paul, et al. “REST and Web Services: In Theory and in Practice.” *REST: from research to practice*. Ed. Erik Wilde and Cesare Pautasso. Springer, 2011. 35–57.

Al-Masri, Eyhab and Qusay H Mahmoud. “Qos-based discovery and ranking of web services.” *Computer Communications and Networks, 2007. ICCCN 2007. Proceedings of 16th International Conferenceon*. IEEE, 2007. 529–534.

Alamri, Atif, Mohamad Eid, and Abdulmotaleb El Saddik. “Classification of the state-of-the-art dynamic web services composition techniques.” *International Journal of Web and Grid Services* 2.2 (2006): 148–166.

Alarcon, Rosa, Erik Wilde, and Jesus Bellido. “Hypermedia-driven RESTful service composition.” *Service-Oriented Computing*. Springer, 2010. 111–120.

Alice, P Sheba, AM Abirami, and A Askarunisa. “An Enhanced Method for Efficient Information Retrieval from Resume Documents using SPARQL.” *Data Mining and Knowledge Engineering* 4.1 (2012): 5–10.

Alrifai, Mohammad and Thomas Risse. “Combining global optimization with local selection for efficient QoS-aware service composition.” *Proceedings of the 18th international conference on World wide web*. ACM, 2009. 881–890.

Alrifai, Mohammad, Thomas Risse, and Wolfgang Nejdl. “A hybrid approach for efficient Web service composition with end-to-end QoS constraints.” *ACM Transactions on the Web (TWEB)* 6.2 (2012): 7.

AlShahwan, Feda, Klaus Moessner, and Francois Carrez. “Evaluation of Distributed SOAP and RESTful MobileWeb Services.” *International Journal on Advances in Networks and Services* 3.3 & 4 (2010): 447–461.

Andry, Francois, Lin Wan, and Daren Nicholson. “A Mobile Application Accessing PatientsHealth Records Through a REST API.” *HEALTHINF 2011 International Conference on Health Informatics*. 2011. 27–32.

Ardagna, Danilo and Barbara Pernici. “Adaptive service composition in flexible processes.” *Software Engineering, IEEE Transactions on* 33.6 (2007): 369–384.

[Ardagna, Danilo and Barbara Pernici. "Global and local QoS constraints guarantee in web service selection." *Web Services, 2005. ICWS 2005. Proceedings. 2005 IEEE International Conference on*. IEEE], 2005.

Arkin, Assaf, et al. "Web service choreography interface (WSCI) 1.0." *Standards proposal by BEA Systems, Intalio, SAP, and Sun Microsystems* (2002).

Armbrust, Michael, et al. "A view of cloud computing." *Communications of the ACM* 53.4 (2010): 50–58.

Arroyo, S, M Stollberg, and Y Ding. "WSMO primer." *DERI Working Draft v01* (2004).

Atzori, Luigi, Antonio Iera, and Giacomo Morabito. "The internet of things: A survey." *Computer networks* 54.15 (2010): 2787–2805.

Badidi, Elarbi. "A Framework for software-as-a-service selection and provisioning." *preprint arXiv:1306.1888* (2013).

Bauer, Florian and Martin Kaltenböck. "Linked open data: The essentials." *Edition mono/monochrom, Vienna* (2011).

Bellur, Umesh, Amit Gupta, and Harin Vadodaria. *Semantic matchmaking algorithms*. INTECH Open Access Publisher, 2008.

Bellur, Umesh and Roshan Kulkarni. "Improved matchmaking algorithm for semantic web services based on bipartite graph matching." *Web Services, 2007. ICWS 2007. IEEE International Conference on*. IEEE, 2007. 86–93.

Bellwood, Tom, et al. "The universal description, discovery and integration (uddi) specification." *Rapport technique, Comité OASIS* (2002).

Benatallah, Boualem, Marlon Dumas, and Zakaria Maamar. "Definition and execution of composite web services: The self-serv project." *IEEE Data Engineering Bulletin* 25.4 (2002): 47–52.

Benatallah, Boualem, et al. "Declarative composition and peer-to-peer provisioning of dynamic web services." *Data Engineering, 2002. Proceedings. 18th International Conference on*. IEEE, 2002. 297–308.

[Benatallah, Boualem, et al. "On automating web services discovery." *The VLDB Journal* 14.1 (2005): 84–96.

Bennara, Mahdi, Michael Mrissa, and Youssef Amghar. "An approach for composing RESTful linked services on the web." *Proceedings of the companion publication of the 23rd international conference on World wide web companion*. International World Wide Web Conferences Steering Committee, 2014. 977–982.

Berners-Lee, Tim, Christian Bizer, and Tom Heath. "Linked data-the story so far." *International Journal on Semantic Web and Information Systems* 5.3 (2009): 1–22.

Berners-Lee, Tim, James Hendler, Ora Lassila, et al. "The semantic web." *Scientific american* 284.5 (2001): 28–37.

Biron, Paul V and Ashok Malhotra. “XML schema part 2: Datatypes second edition. W3C recommendation.” *World Wide Web Consortium (W3C), Oct* 28 (2004): 110–114.

Bohara, Mohammed Husain, Mahesh K Mishra, and Shubham Chaudhary. “RESTful Web Service Integration Using Android Platform.” *Computing, Communications and Networking Technologies (ICCCNT), 2013 Fourth International Conference on*. IEEE, 2013. 1–6.

Booth, David, et al. “Web services architecture.” (2004).

Bose, Sumit, et al. “SLA management in Cloud Computing: A service provider?s perspective.” *Cloud Computing: Principles and paradigms, Rajkumar Buyya, James Broberg, and Andrzej Goscinski, Eds. New Jersey, USA: John Wiley & Sons* (2011): 413–436.

Bray, Tim, et al. “Extensible markup language (XML).” *World Wide Web Consortium Recommendation REC-xml-19980210. http://www. w3. org/TR/1998/REC-xml-19980210* 16 (1998).

Brickley, Dan and Ramanathan V Guha. “Resource Description Framework (RDF) Schema Specification 1.0: W3C Candidate Recommendation 27 March 2000.” (2000).

Broens, Tom, et al. “Context-aware, ontology-based service discovery.” *Ambient Intelligence*. Springer, 2004. 72–83.

Bukhari, Ahmad C and Christopher JO Baker. “The Canadian health census as Linked Open Data: towards policy making in public health.” *9th International Conference on Data Integration in the Life Sciences; July 11-12, 2013; Montreal, PQ URL: http://www2. unb. ca/csas/data/ws/dils2013/papers/DILS-SYS-EC-paper3. pdf*. 2013.

Buyya, Rajkumar, et al. “Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility.” *Future Generation computer systems* 25.6 (2009): 599–616.

Calheiros, Rodrigo N, et al. “CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms.” *Software: Practice and Experience* 41.1 (2011): 23–50.

Canfora, Gerardo, et al. “An approach for QoS-aware service composition based on genetic algorithms.” *Proceedings of the 7th annual conference on Genetic and evolutionary computation*. ACM, 2005. 1069–1075.

Cardoso, Jorge, A Sheth, and Liyang Yu. “Semantic Web services, processes and applications.” *RECIIS* 3.1 (2009): 85–88.

Carman, Mark, Luciano Serafini, and Paolo Traverso. “Web service composition as planning.” *ICAPS 2003 workshop on planning for web services*. 2003. 1636–1642.

“[Cases of Influenza A H1N1 (Swine Flu) State/UT wise, Year wise for 2009, 2010, 2011 and 2012.” Online; accessed: 2015-11-30].

Cavalcante, Everton, et al. “Cloud Integrator: Building value-added services on the cloud.” *Network Cloud Computing and Applications (NCCA), 2011 First International Symposium on*. IEEE, 2011. 135–142.

Chandrasekaran, Balakrishnan, John R Josephson, and V Richard Benjamins. “What are ontologies, and why do we need them?” *IEEE Intelligent systems* 1 (1999): 20–26.

Choi, Okkyung, Sangyong Han, and Ajith Abraham. “Extended semantic web services model for automatic integrated framework.” *Next Generation Web Services Practices, 2005. NWeSP 2005. International Conference on*. IEEE, 2005. 8–pp.

Da Silva, Eduardo Gonçalves, Luís Ferreira Pires, and Marten Van Sinderen. “Towards runtime discovery, selection and composition of semantic services.” *Computer communications* 34.2 (2011): 159–168.

De Oliveira Jr, Frederico G Alvares and José M Parente de Oliveira. “QoS-based Approach for Dynamic Web Service Composition.” *Journal of Universal Computer Science* 17.5 (2011): 712–741.

Della Valle, Emanuele and Dario Cerizza. “Cocoon glue: a prototype of wsmo discovery engine for the healthcare field.” *Proceedings of 2nd WSMO Implementation Workshop WIW*. 2005.

Della Valle, Emanuele, et al. “The need for semantic web service in the eHealth.” *W3C workshop on Frameworks for Semantics in Web Services*. 2005.

Dillon, Tharam, Chen Wu, and Elizabeth Chang. “Cloud computing: issues and challenges.” *Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on*. IEEE, 2010. 27–33.

Dogac, Asuman, et al. “Artemis: Deploying semantically enriched Web services in the healthcare domain.” *Information Systems* 31.4 (2006): 321–339.

Drago, Idilio. *Understanding and monitoring cloud services*. University of Twente, 2013.

Dustdar, Schahram and Wolfgang Schreiner. “A survey on web services composition.” *International journal of web and grid services* 1.1 (2005): 1–30.

Eberhart, Russ C and James Kennedy. “A new optimizer using particle swarm theory.” *Proceedings of the sixth international symposium on micro machine and human science*. New York, NY, 1995. 39–43.

Erl, Thomas. *Service-oriented architecture: concepts, technology, and design*. Pearson Education India, 2005.

Fadel, Ronald. “Planning with complex actions.” Diss. stanford university, 2002.

Fanjiang, Yong-Yi, et al. “Genetic algorithm for QoS-aware dynamic web services composition.” *Machine Learning and Cybernetics (ICMLC), 2010 International Conference on*. IEEE, 2010. 3246–3251.

Fensel, Dieter and Christoph Bussler. “The web service modeling framework WSMF.” *Electronic Commerce Research and Applications* 1.2 (2002): 113–137.

Fethallah, Hadjila, et al. “QoS-aware service selection based on swarm particle optimization.” *Information Technology and e-Services (ICITeS), 2012 International Conference on*. IEEE, 2012. 1–6.

Fielding, Roy Thomas. “Architectural styles and the design of network-based software architectures.” Diss. University of California, Irvine, 2000.

Forgy, Charles L. “Rete: A fast algorithm for the many pattern/many object pattern match problem.” *Artificial intelligence* 19.1 (1982): 17–37.

Fujii, Keita and Tatsuya Suda. “Dynamic service composition using semantic information.” *Proceedings of the 2nd international conference on Service oriented computing*. ACM, 2004. 39–48.

García, Andrés García and Ignacio Blanquer. “Cloud Services Representation using SLA Composition.” *Journal of Grid Computing* 13.1 (2015): 35–51.

Garg, Saurabh Kumar, Steve Versteeg, and Rajkumar Buyya. “A framework for ranking of cloud computing services.” *Future Generation Computer Systems* 29.4 (2013): 1012–1023.

Gholam Hassan Tabatabaei, Sayed, et al. “Security conscious AI-planning-based composition of semantic web services.” *International Journal of Web Information Systems* 6.3 (2010): 203–229.

Gibb, Brian K and Suresh Damodaran. *ebXML: Concepts and application*. John Wiley & Sons, Inc., 2002.

Godse, Manish and Shrikant Mulik. “An approach for selecting software-as-a-service (SaaS) product.” *Cloud Computing, 2009. CLOUD’09. IEEE International Conference on*. IEEE, 2009. 155–158.

Gomadam, Karthik, Ajith Ranabahu, and Amit Sheth. “Sa-rest: Semantic annotation of web resources.” *W3C Member Submission* 5 (2010): 52.

Gomadam, Karthik, et al. “A declarative approach using SAWSDL and semantic templates towards process mediation.” *Semantic Web Services Challenge*. Springer, 2009. 101–118.

Gruber, Thomas R. “A translation approach to portable ontology specifications.” *Knowledge acquisition* 5.2 (1993): 199–220.

Gudgin, Martin, et al. “Soap version 1.2 part 1: Messaging framework.” *W3C Working Draft, DevelopMentor, Sun, IBM, Canon, Microsoft* (2002).

Guidi, Claudio, Saverio Giallorenzo, and Maurizio Gabbrielli. “Towards a composition-based APIaaS layer.” *CLOSER 2014*. 2015.

Gustavo, Alonso, et al. *Web services: concepts, architectures and applications*. Springer, 2004.

Gutierrez-Garcia, J Octavio and Kwang Mong Sim. “Agent-based cloud service composition.” *Applied intelligence* 38.3 (2013): 436–464.

Hadley, Marc J. “Web application description language (WADL).” (2006).

Han, Taekgyeong and Kwang Mong Sim. “An ontology-enhanced cloud service discovery system.” *Proceedings of the International MultiConference of Engineers and Computer Scientists*. 2010. 17–19.

Hatzi, Ourania, et al. “An integrated approach to automated semantic web service composition through planning.” *Services Computing, IEEE Transactions on* 5.3 (2012): 319–332.

Hepp, Martin and Dumitru Roman. “An ontology framework for semantic business process management.” *Wirtschaftinformatik Proceedings 2007* (2007): 27.

Höing, André. “Orchestrating secure workflows for cloud and grid services.” Diss. Berlin Institute of Technology, 2010.

Hristoskova, Anna, et al. “Dynamic composition of medical support services in the ICU: Platform and algorithm design details.” *Computer methods and programs in biomedicine* 100.3 (2010): 248–264.

Huang, Lican, et al. “A Qos Optimization for Intelligent and Dynamic Web Service Composition Based on Improved PSO Algorithm.” *Networking and Distributed Computing (ICNDC), 2011 Second International Conference on*. IEEE, 2011. 214–217.

Ji, Xiang. “Social Data Integration and Analytics for Health Intelligence.” (2014).

John, Davis and MS Rajasree. “Restdoc: Describe, discover and compose restful semantic web services using annotated documentations.” *International Journal of Web & Semantic Technology (IJWesT)* 4.1 (2013).

Joshi, Karuna P, Yelena Yesha, and Tim Finin. “Automating cloud services life cycle through semantic technologies.” *Services Computing, IEEE Transactions on* 7.1 (2014): 109–122.

Jrad, Foued, Jie Tao, and Achim Streit. “SLA based Service Brokering in Intercloud Environments.” *CLOSER* 2012 (2012): 76–81.

Jrad, Foued, et al. “A utility–based approach for customised cloud service selection.” *International Journal of Computational Science and Engineering* 10.1-2 (2015): 32–44.

Karastoyanova, Dimka, et al. “A Reference Architecture for Semantic Business Process Management Systems.” *Multikonferenz Wirtschaftsinformatik*. 2008. 1727–1738.

Karim, Rashed, Chen Ding, and Ali Miri. “An end-to-end QoS mapping approach for cloud service selection.” *Services (SERVICES), 2013 IEEE Ninth World Congress on*. IEEE, 2013. 341–348.

Keet, C Maria. “The use of foundational ontologies in ontology development: an empirical assessment.” *The Semantic Web: Research and Applications*. Springer, 2011. 321–335.

Knoblock, Craig A, et al. “Semi-automatically mapping structured sources into the semantic web.” *The Semantic Web: Research and Applications*. Springer, 2012. 375–390.

Kona, Srividya, et al. “Automatic Composition of Semantic Web Services.” *ICWS*. 2007. 150–158.

[Kona, Srividya, et al. “WSC-2009: a quality of service-oriented web services challenge.” *Commerce and Enterprise Computing, 2009. CEC’09. IEEE Conference on*. IEEE], 2009. 487–490.

Kopecky, Jonathon, Karthik Gomadam, and Tomas Vitvar. “hrests: An html microformat for describing restful web services.” *Web Intelligence and Intelligent Agent Technology, 2008. WI-IAT’08. IEEE/WIC/ACM International Conference on*. IEEE, 2008. 619–625.

Kuo, Mu-Hsing. “Opportunities and challenges of cloud computing to improve health care services.” *Journal of medical Internet research* 13.3 (2011): e67.

Küster, Ulrich, et al. “DIANE: an integrated approach to automated service discovery, matchmaking and composition.” *Proceedings of the 16th international conference on World Wide Web*. ACM, 2007. 1033–1042.

Kuter, Ugur and Dana Nau. “Forward-chaining planning in nondeterministic domains.” *AAAI*. 2004. 513–518.

"Laboratory confirmed Cases and Deaths caused by Pandemic Influenza A H1N1: State/ UTwise." Online; accessed: 2015-11-30.

Lacy, Lee W. *OWL: representing information using the Web Ontology Language*. Trafford Publishing, 2005.

Laliwala, Zakir, et al. "Semantic and rules based Event-Driven dynamic web services composition for automation of business processes." *Services Computing Workshops, 2006. SCW'06. IEEE*. IEEE, 2006. 175–182.

Lassila, Ora and Sapna Dixit. "Interleaving discovery and composition for simple workflows." *in Proceedings of the AAAI Spring Symposium on Semantic Web Services, 22nd-24th March*. 2004.

Lassila, Ora, Ralph R Swick, et al. "Resource Description Framework (RDF) model and syntax specification." (1998).

Lee, Jonathan, Shin-Jie Lee, and Ping-Feng Wang. "A Framework for Composing SOAP, Non-SOAP and Non-Web Services." *Services Computing, IEEE Transactions on* 8.2 (2015): 240–250.

Lee, Yong-Ju. "Semantic-Based Web API Composition for Data Mashups." *Journal of Information Science and Engineering* 31.4 (2015): 1233–1248.

Lee, Yugyung, et al. "Compositional knowledge management for medical services on semantic web." *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*. ACM, 2004. 498–499.

Li, Ning, et al. "OmniVoke: a framework for automating the invocation of Web APIs." *Semantic Computing (ICSC), 2011 Fifth IEEE International Conference on*. IEEE, 2011. 39–46.

[Li, Zheng, et al. "Effort-oriented classification matrix of web service composition." *Internet and Web Applications and Services (ICIW), 2010 Fifth International Conference on*. IEEE], 2010. 357–362.

Liu, Dong. "RESTful Service Composition." Diss. Citeseer, 2013.

Liu, Jiamao, Juntao Cui, and Ning Gu. "Composing web services dynamically and semantically." *E-Commerce Technology for Dynamic E-Business, 2004. IEEE International Conference on*. IEEE, 2004. 234–241.

Liu, Li, et al. "Ontology-based service matching in cloud computing." *Fuzzy Systems (FUZZ-IEEE), 2014 IEEE International Conference on*. IEEE, 2014. 2544–2550.

Liu, Yutu, Anne H Ngu, and Liang Z Zeng. "QoS computation and policing in dynamic web service selection." *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*. ACM, 2004. 66–73.

Ma, Hui, Anqi Wang, and Mengjie Zhang. "A Hybrid Approach Using Genetic Programming and Greedy Search for QoS-Aware Web Service Composition." *Transactions on Large-Scale Data-and Knowledge-Centered Systems XVIII*. Springer, 2015. 180–205.

Mabrouk, Nebil Ben, et al. "QoS-aware service composition in dynamic service oriented environments." *Middleware 2009*. Ed. JeanM. Bacon and BrianF. Cooper. Vol. 5896. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2009. 123–142.

Maleshkova, Maria, Carlos Pedrinaci, and John Domingue. "Semantic annotation of Web APIs with SWEET." (2010).

Mallayya, Deivamani, Baskaran Ramachandran, and Suganya Viswanathan. "An Automatic Web Service Composition Framework Using QoS-Based Web Service Ranking Algorithm." *The Scientific World Journal* 2015 (2015).

Manola, Frank, Eric Miller, and B McBride. "RDF Primer, W3C Recommendation 10 February 2004, 2004." *WWW: http://www. w3. org/TR/2004/REC-rdf-primer-20040210/(Februar 2004)* 19 ().

Martin, David, et al. "Bringing semantics to web services: The OWL-S approach." *Semantic Web Services and Web Process Composition*. Springer, 2005. 26–42.

Maximilien, E Michael and Munindar P Singh. "Toward autonomic web services trust and selection." *Proceedings of the 2nd international conference on Service oriented computing*. ACM, 2004. 212–221.

McDermott, Drew. "Estimated-regression planning for interactions with web services." *Proceedings of the Sixth International Conference on Artificial Intelligence Planning Systems*. 2002.

McGuinness, Deborah L, Frank Van Harmelen, et al. "OWL web ontology language overview." *W3C recommendation* 10.10 (2004).

McIlraith, Sheila and Tran Cao Son. "Adapting golog for composition of semantic web services." *KR* 2 (2002): 482–493.

[McIlraith, Sheila and Tran Cao Son. "Adapting Golog for programming the semantic web." *Fifth International Symposium on Logical Formalizations of Commonsense Reasoning*. 2001]. 195–202.

McIlraith, Sheila A, Tran Cao Son, and Honglei Zeng. "Semantic web services." *IEEE intelligent systems* 2 (2001): 46–53.

Medjahed, Brahim and Athman Bouguettaya. *Service composition for the Semantic Web*. Springer Science & Business Media, 2011.

Medjahed, Brahim, Athman Bouguettaya, and Ahmed K Elmagarmid. "Composing web services on the semantic web." *The VLDB Journal* 12.4 (2003): 333–351.

Mell, Peter and Tim Grance. "The NIST definition of cloud computing." (2011).

Menzel, Michael, Marten Schönherr, and Stefan Tai. "(MC2) 2: criteria, requirements and a software prototype for Cloud infrastructure decisions." *Software: Practice and Experience* 43.11 (2013): 1283–1297.

Michlmayr, Anton, et al. "End-to-end support for qos-aware service selection, binding, and mediation in vresco." *Services Computing, IEEE Transactions on* 3.3 (2010): 193–205.

Milanovic, Nikola and Miroslaw Malek. "Current solutions for web service composition." *IEEE Internet Computing* 6 (2004): 51–59.

Miller, John, et al. “Wsdl-s: Adding semantics to wsdl-white paper.” *University of Georgia* (2004).

Modi, Kirit J and Sanjay Garg. “Dynamic Web Services Composition using Optimization Approach.” *International Journal of Computer Science & Communication* 6.2 (2015): 285–293.

Moghaddam, Mahboobeh and Joseph G Davis. “Service selection in web service composition: A comparative review of existing approaches.” *Web Services Foundations*. Ed. Athman Bouguettaya, Quan Z. Sheng, and Florian Daniel. Springer New York, 2014. 321–346.

Mouhoub, Mohamed Lamine, Daniela Grigori, and Maude Manouvrier. “A Framework for Searching Semantic Data and Services with SPARQL.” *Service-Oriented Computing*. Springer, 2014. 123–138.

Mukhopadhyay, Debajyoti and Archana Chougule. “A survey on web service discovery approaches.” *Advances in Computer Science, Engineering & Applications*. Springer, 2012. 1001–1012.

Neupane, Sanjeev, et al. “Integrating Heterogeneous Web Services into a Seamless Application.” *Int. J. on Recent Trends in Engineering & Technology* 5.01 (2011): 149–152.

Newcomer, Eric. *Understanding Web Services: XML, Wsdl, Soap, and UDDI*. Addison-Wesley Professional, 2002.

[Ngan, Le Duy and Rajaraman Kanagasabai. “Owl-s based semantic cloud service broker.” *Web Services (ICWS), 2012 IEEE 19th International Conference on*. IEEE], 2012. 560–567.

Ngan, Le Duy and Rajaraman Kanagasabai. “Semantic Web service discovery: state-of-the-art and research challenges.” *Personal and ubiquitous computing* 17.8 (2013): 1741–1752.

Ouzzani, Mourad and Athman Bouguettaya. “Efficient access to web services.” *Internet Computing, IEEE* 8.2 (2004): 34–44.

Paganelli, Federica, et al. “An information-centric and REST-based approach for EPC Information Services.” *Journal of Communications Software & Systems* 9.1 (2013).

Page, Kevin R, David C De Roure, and Kirk Martinez. “REST and Linked Data: a match made for domain driven development?” *Proceedings of the Second International Workshop on RESTful Design*. ACM, 2011. 22–25.

Paikari, Elham, et al. “Multi-Agent system for semantic web service composition.” *Knowledge Science, Engineering and Management*. Springer, 2011. 305–317.

Papazoglou, Michael P, et al. “Service-oriented computing: a research roadmap.” *International Journal of Cooperative Information Systems* 17.02 (2008): 223–255.

[Papazoglou, Michael P, et al. “Service-oriented computing: State of the art and research challenges.” *Computer* 40.11 (2007): 38–45.

Papazoglou, Mike P and Willem-Jan Van Den Heuvel. “Service oriented architectures: approaches, technologies and research issues.” *The VLDB journal* 16.3 (2007): 389–415.

Parekh, Maulik and B Saleena. “Designing a Cloud Based Framework for Health Care System and Applying Clustering Techniques for Region Wise Diagnosis.” *Procedia Computer Science* 50 (2015): 537–542.

Parra, Jorge, et al. “RESTful Discovery and Eventing for Service Provisioning in Assisted Living Environments.” *Sensors* 14.5 (2014): 9227–9246.

Pautasso, Cesare. “RESTful Web service composition with BPEL for REST.” *Data & Knowledge Engineering* 68.9 (2009): 851–866.

Pautasso, Cesare, Olaf Zimmermann, and Frank Leymann. “Restful Web Services vs. ”Big”’ Web Services: Making the Right Architectural Decision.” *Proceedings of the 17th international conference on World Wide Web*. WWW ’08. ACM, 2008. 805–814.

Pedrinaci, Carlos, et al. “iServe: a linked services publishing platform.” *CEUR workshop proceedings*. 2010.

Peltz, Chris. “Web services orchestration and choreography.” *Computer* 10 (2003): 46–52.

Pistore, Marco, et al. “Planning and monitoring web service composition.” *Artificial Intelligence: Methodology, Systems, and Applications*. Springer, 2004. 106–115.

Ponnekanti, Shankar R and Armando Fox. “Sword: A developer toolkit for web service composition.” *Proc. of the Eleventh International World Wide Web Conference, Honolulu, HI*. 2002.

PrudH́ommeaux, Eric, Andy Seaborne, et al. “SPARQL query language for RDF.” *W3C recommendation* 15 (2008).

Rajeswari, M, et al. “Appraisal and analysis on various web service composition approaches based on QoS factors.” *Journal of King Saud University-Computer and Information Sciences* 26.1 (2014): 143–152.

Rao, Jinghai and Xiaomeng Su. “A survey of automated web service composition methods.” *Semantic Web Services and Web Process Composition*. Springer, 2005. 43–54.

Richardson, Leonard and Sam Ruby. *RESTful web services*. ” O’Reilly Media, Inc.”, 2008.

Rodriguez Mier, Pablo, et al. “An Integrated Semantic Web Service Discovery and Composition Framework.” *Services Computing, IEEE Transactions on* PP.99 (2015): 1–1.

Saquicela, Victor, Luis M Vilches-Blázquez, and Óscar Corcho. *Semantic Annotation of RESTful services using external resources*. Springer, 2010.

Sathya, M, et al. “Evaluation of qos based web-service selection techniques for service composition.” *International Journal of Software Engineering* 1.5 (2010): 73–90.

Shapiro, Robert M. “XPDL 2.0: Integrating process interchange and BPMN.” *Workflow Handbook* (2006): 183–194.

Shawish, Ahmed and Maria Salama. “Cloud Computing: Paradigms and Technologies.” *Inter-cooperative Collective Intelligence: Techniques and Applications*. Ed. Fatos Xhafa and Nik Bessis. Vol. 495. Studies in Computational Intelligence. Springer, 2014. 39–67.

Shehu, Umar, Gregory Epiphaniou, and Ghazanfar Ali Safdar. “A survey of QoS-aware web service composition techniques.” *International Journal of Computer Applications* 89.12 (2014).

Sheng, Quan Z, et al. “Web services composition: A decadeś overview.” *Information Sciences* 280 (2014): 218–238.

Sheth, Amit, Pramod Anantharam, and K Thirunarayan. “kHealth: Proactive Personalized Actionable Information for Better Healthcare.” *Workshop on Personal Data Analytics in the Internet of Things (PDA@ IOT 2014), collocated at VLDB*. 2014.

Siegel, Jane and Jeff Perdue. “Cloud services measures for global use: the Service Measurement Index (SMI).” *SRII Global Conference (SRII), 2012 Annual*. IEEE, 2012. 411–415.

Sim, Kwang Mong. “Agent-based cloud computing.” *Services Computing, IEEE Transactions on* 5.4 (2012): 564–577.

Singh, Munindar P and Michael N Huhns. *Service-oriented computing: semantics, processes, agents*. John Wiley & Sons, 2006.

Sirin, Evren, James Hendler, and Bijan Parsia. “Semi-automatic composition of web services using semantic descriptions.” *1st Workshop on Web Services: Modeling, Architecture and Infrastructure*. 2003. 17–24.

Sirin, Evren, et al. “HTN planning for web service composition using SHOP2.” *Web Semantics: Science, Services and Agents on the World Wide Web* 1.4 (2004): 377–396.

Soman, AK. *Cloud-based Solutions for Healthcare IT*. CRC Press, 2011.

[Speiser, Sebastian and Andreas Harth. “Integrating linked data and services with linked data services.” *The Semantic Web: Research and Applications*. Springer], 2011. 170–184.

Speiser, Sebastian and Andreas Harth. “Towards linked data services.” *Proceedings of the 9th International Semantic Web Conference (ISWC)*. Citeseer, 2010.

Stadtmüller, Steffen, et al. “Data-fu: A language and an interpreter for interaction with read/write linked data.” *Proceedings of the 22nd international conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2013. 1225–1236.

Standard, OASIS. “Web services business process execution language version 2.0.” *URL: http://docs. oasis-open. org/wsbpel/2.0/OS/wsbpel-v2. 0-OS. html* (2007).

Sun, Le, et al. “Cloud service selection: State-of-the-art and future research directions.” *Journal of Network and Computer Applications* 45 (2014): 134–150.

Talantikite, Hassina Nacer, Djamil Aissani, and Nacer Boudjlida. “Semantic annotations for web services discovery and composition.” *Computer Standards & Interfaces* 31.6 (2009): 1108–1117.

Tang, Maolin and Lifeng Ai. “A hybrid genetic algorithm for the optimal constrained web service selection problem in web service composition.” *Evolutionary Computation (CEC), 2010 IEEE Congress on*. IEEE, 2010. 1–8.

Thatte, Satish. “XLANG: Web services for business process design.” *Microsoft Corporation* (2001): 13.

Thiagarajan, Rajesh K, et al. “BPML: a process modeling language for dynamic business models.” *Advanced Issues of E-Commerce and Web-Based Information Systems, 2002.(WECWIS 2002). Proceedings. Fourth IEEE International Workshop on*. IEEE, 2002. 222–224.

Tilahun, Binyam, et al. "Design and development of a linked open data-based health information representation and visualization system: Potentials and preliminary evaluation." *JMIR medical informatics* 2.2 (2014).

Ummel, Ed Mitchell. "The Rise of the Semantic Enterprise." *Cutter IT Journal* 22.9 (2009): 01.

Unhelkar, B and S Murugesan. "Accruing Business Value Through the Adoption of Semantic Web Technologies." *Cutter IT Journal* 22.9 (2009): 24–30.

Upadhyaya, Bipin. "Composing Heterogeneous Services From End Users' Perspective." (2014).

Verborgh, Ruben, et al. "Description and interaction of RESTful services for automatic discovery and execution." *Proceedings of the FTRA 2011 International Workshop on Advanced Future Multimedia Services*. 2011.

Vu, Le-Hung, Manfred Hauswirth, and Karl Aberer. "QoS-based service selection and ranking with trust and reputation management." *On the move to meaningful internet systems 2005: CoopIS, DOA, and ODBASE*. Springer, 2005. 466–483.

Vuković, Maja, Evangelos Kotsovinos, and Peter Robinson. "An architecture for rapid, on-demand service composition." *Service Oriented Computing and Applications* 1.4 (2007): 197–212.

Wang, Shangguang, et al. "Cloud model for service selection." *Computer Communications Workshops (INFOCOM WKSHPS), 2011 IEEE Conference on*. IEEE, 2011. 666–671.

[Wang, Shuying, et al. "A process oriented semantic healthcare service composition." *Science and Technology for Humanity (TIC-STH), 2009 IEEE Toronto International Conference*. IEEE], 2009. 479–484.

Weise, Thomas, et al. "Different approaches to semantic web service composition." *Internet and Web Applications and Services, 2008. ICIW'08. Third International Conference on*. IEEE, 2008. 90–96.

Yang, Jian and Mike P Papazoglou. "Service components for managing the life-cycle of service compositions." *Information Systems* 29.2 (2004): 97–125.

Yang, Stephen JH, Jia Zhang, and Blue CW Lan. "Service-level agreement-based QoS analysis for web services discovery and composition." *International Journal of Internet and Enterprise Management* 5.1 (2006): 39–58.

Ye, Zhen, Athman Bouguettaya, and Xiaofang Zhou. "QoS-aware cloud service composition based on economic models." *Service-Oriented Computing*. Ed. Chengfei Liu, et al. Vol. 7636. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2012. 111–126.

Ye, Zhen, et al. "Long-term QoS-aware Cloud Service Composition using Multivariate Time Series Analysis." *Services Computing, IEEE Transactions on* PP.99 (2014): 1–1.

Yoon, K Paul and Ching-Lai Hwang. *Multiple attribute decision making: an introduction*. Vol. 104. Sage publications, 1995.

Yu, Qi and Athman Bouguettaya. *Foundations for efficient web service selection*. Springer Science & Business Media, 2009.

Yu, Tao, Yue Zhang, and Kwei-Jay Lin. "Efficient algorithms for Web services selection with end-to-end QoS constraints." *ACM Transactions on the Web (TWEB)* 1.1 (2007): 6.

[Zeng, Liangzhao, et al. "Qos-aware middleware for web services composition." *Software Engineering, IEEE Transactions on* 30.5 (2004): 311–327.

Zeng, Liangzhao, et al. "Quality driven web services composition." *Proceedings of the 12th international conference on World Wide Web*. WWW '03. ACM, 2003. 411–421.

Zhang, Miranda, et al. "A Declarative Recommender System for Cloud Infrastructure Services Selection." *GECON*. Springer, 2012. 102–113.

Zhang, Ruoyan, Ismailcem Budak Arpinar, and Boanerges Aleman-Meza. "Automatic Composition of Semantic Web Services." *ICWS*. 2003. 38–41.

[Zhang, Xutang, et al. "Ontology-based semantic retrieval for engineering domain knowledge." *Neurocomputing* 116 (2013): 382–391.

Zhang, Ying, et al. "A semantic approach to retrieving, linking, and integrating heterogeneous geospatial data." *Joint Proceedings of the Workshop on AI Problems and Approaches for Intelligent Environments and Workshop on Semantic Cities*. ACM, 2013. 31–37.

Zhao, Haibo and Prashant Doshi. "Towards automated RESTful web service composition." *Web Services, 2009. ICWS 2009. IEEE International Conference on*. IEEE, 2009. 189–196.

Zunino, Alejandro and Marcelo Campo. "A Survey of Approaches to Web Service Discovery in Service Oriented Architectures." *Innovations in Database Design, Web Applications, and Information Systems Management* (2012): 107.