Addis Ababa University

College of Natural Sciences

Semantic Web Service Matchmaking Model

Nigussie Seid Mohammed

A Thesis Submitted to the Department of Computer Science in Partial Fulfillment for the Degree of Master of Science in Computer Science

Addis Ababa, Ethiopia
Jun 2018

Addis Ababa University
College of Natural Sciences

Nigussie Seid Mohammed

Advisor: *Mulugeta Libsie (PhD)*

This is to certify that the thesis prepared by *Nigussie Seid*, titled: *Semantic Web Service Matchmaking Model* and submitted in partial fulfillment of the requirements for the Degree of Master of Science in Computer Science complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the Examining Committee:

Name Signature ____ Date

Advisor: _____

Examiner: _____

Examiner: _____

Abstract

The success of the Web and its accumulated huge information demands for new technologies to enable machines to understand and act by their own assisting the human user. The Semantic Web technologies emerge as an effort to transform the current syntactically interconnected data into a semantically and meaningful linked data that enable machine to understand and act by their own. The semantic Web service initiative is a similar effort working on transforming the widely used and adopted distributed computing technology, the Web Service.

Since its introduction, the Web Service technology enjoyed wide adoption as a technology of interoperability among diversified technological platforms and entities. Its neutrality, open and standard based framework make it favorable. Despite its wide usage, it remains verily syntactical demanding intensive human intervention for matching a Consumer's service request to the Provider's service offerings. With the help of the semantic technologies, the hybrid Semantic Web Service is attempting to fully or partially automate this process.

Matchmaking is a process of finding the most appealing service, out of the existing many offerings by Providers, that can render the functionalities required by the Consumer. It is a complex process that requires careful analysis of the Consumer's request against Providers' service offerings. Our model introduces special Service Ontologies, Consumer data and similarity measure computation formulae to the Web Service architecture. A specialized service similarity measure computation is introduced which considers the special nature of services. It computes similarity of services based on a composite of service name, output and result leaving aside inputs for negotiation. Unlike the traditional matchmaking process, the matchmaking process is split between the Service Consumer and Broker in which the Broker matches the user query to identify candidate services while the Consumer makes negotiation with the candidates to make final selection. The service inputs are determined during the negotiation process, as the Consumer would not know it ahead of time. In addition, the Service Ontology is a specialized domain Ontology that details the domain's possible services, results, condition, inputs and outputs including their similarity and dependence with each other.

Keywords: Semantic Web Service, Web Service, Semantic Web, Web Service Matchmaking

Acknowledgments

My special thanks goes to my family (my wife Zinet and my sons – Imran and Ihsan) and friends, who were behind me all the time supporting and nagging me to get this work done. My advisor, Dr. Mulugeta, I sincerely thank you for your support and patience, to get me here throughout this long time and my excuses. And finally, praise is to the Almighty Allah.

Table of Contents

Acronyms and Abbreviations

DDOS	Distributed Denial of Service
DOS	Denial of Service
FLOWS	First-order Logic Ontology for Web Services
IOP	Input, Output and Precondition
IOPR	Input, Output, Precondition and Result
LCS	Least Common Subsummer
LSLO	Local Selection and Local Optimization
MORP	Mixed Opinion based Reputation Prediction
NLP	Natural Language Processing
NSA	Negotiating Software Agents
OWL-S	Web Ontology Language for Services
QoS	Quality of Service
QoWSO	Quality of Web Service Description Ontology
SAWSDL	Semantic Annotation for Web Service Description Language
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SUMO	Suggested Upper Merged Ontology
SWSA	Semantic Web Services Initiative Architecture
SWSD	Semantic Web Service Description
SWSO	Semantic Web Services Ontology
UDDI	Universal Description, Discovery and Integration
URR	Usage Restriction Rate
W3C	World Wide Web Concretum
WSDL	Web Service Description Language
WSDL-S	Web Service Description Language Semantics
WSML	Web Service Modeling Language
WSMO	Web Service Modeling Ontology
WSO	Web Service Ontology
WWW	World Wide Web
XML	Extensible Markup Language

List of Tables

List of Figures

List of Algorithms

Chapter One: Introduction

1.1 Overview

The need for application-to-application and cross platform communication and information exchange has significantly grown in parallel with the success of the World Wide Web (WWW) [1]. Service Oriented Architecture (SOA) enabled by Web Services has received wide acceptance in bridging the demand by providing a standard and modular communication among different software applications, running on a variety of platforms and/or frameworks [2].

Web Services enable standard, cross platform and modular development and deployment of application modules that can be consumed by any interested user/application [1]. According to World Wide Web Concretum's (W3C) definition "a web service is a software system identified by a URI, whose public interfaces and bindings are defined and described using XML. Its 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" [1]

On the other hand, semantic Web which is recognized as the next generation Web provides semantics to the existing Web. It is hoped to enable machines understanding and processing to the existing mostly human understandable and syntactic Web [3]. In an effort to realize the next generation Web the two most recent and vibrant technologies – Semantic Web and Web Service, have joined to overcome the existing limitations forming Semantic Web Service [3, 4].

The success of Web Service came from its modularity, standards and platform independence [1]. The Web service matchmaking process is either fully manual or need great deal of human intervention [3, 5]. Significant research efforts are undergoing to fully automate the matchmaking process through incorporating semantic Web techniques [3, 6]. This effort has incorporated some level of semantics for the existing pure syntactical techniques.

In recent years various models, frameworks and languages are being developed to support the Semantic Web Service [7,8]. The field is infant and needs significant improvement to attain the aspired goal. In this thesis a sematic Web Service matchmaking model is proposed. This model attempts to bridge the limitation of current matchmaking techniques.

1.2 Motivation

The founder of the World Wide Web (WWW), Tim Berners-Lee, envisioned the next generation Web to enable machines and humans work in collaboration. The existing human understandable and syntactic Web will turn into a repository of large interconnected data that can be understood by both humans and machines [9]. This will enable machines to undertake sophisticated and intelligent activities on behalf of humans with little or no supervision.

The current Web is far from the vision of its inventor. Still the Web is dominantly syntactic and embedding semantics is an undergoing effort. Researchers are convinced that the next Web will be realised when standard and interoperable methods are developed that will enable applications to interact with other applications crossing any compatibility issues [10]. XML and Web services were solutions for standardization and interoperability and the addition of semantic on these technologies was another success.

The scenario is, if a firm wishes to have a weather forecast information on its Web site, it could implement the service from the BBC. Given the need of the firm is just to have weather information, its needs can be fulfilled by any weather information service provider, not only from the BBC. In case the BBC service fails to satisfy the request, there should be an easy and straight forward approach to find and use (without the intervention of the developer) some other weather forecast service provider from the Web, may be from Yahoo or CNN.

Despite the set of technologies and techniques developed so far, it is almost impossible to achieve the above scenario without human intervention. Current techniques are highly manual and need direct human (developer) intervention to execute these activities [3, 5].

To enable computers act by their own, there need to be a proven model for automatic service selection and matching among other requirements. The next generation Web's promises are inspiring and interesting to be part of.

1.3 Research problem

In the last decade many companies (commercial, Public, Government or any kind) are publishing their services on the Web using Web Service technologies [7]. Its modularity, standards and platform independence account the major reasons for its wide acceptance [1]. In parallel to its wide

and cross industry adaptation the need for automatic discovery, matching, composition and execution is becoming critical [3, 5, 7].

Currently, developers need to manually discover and analyze Web Services. The developer needs to decide if a Web service satisfies his/her requirement and needs to manually code the matching and execution [5]. Service consumers are not capable of automatically discovering, matching and executing services. If one service provider fails to provide the service or relocate the service, the consumer will not be able to automatically find at the relocated destination or will not find any other service provider (even if there are other service providers with same requirement and functionality) [8].

The semantic Web service comes in to play and incorporate semantic Web technologies in Web services to give semantics to the traditional syntactic Web service [4]. Researchers have resulted in the development of a number of formal languages like Web Ontology Language for Services (OWL-S), Web Service Description Language Semantics (WSDL-S), Web Service Modeling Language (WSML), etc., that are aimed at semantically annotating Web services [5].

The goal of semantic Web service is to enable applications and software Agents to automatically discover, match, compose and execute Services available on the Web to achieve the user's request and significantly eliminating the need of human intervention in the process [7]. Semantic Web services are the enablers of the next generation Web in which machines and humans work in cooperation to achieve a common goal [9].

So far, the Semantic Web Services development is tightly attached to the semantic description languages like OWL-S, Web Service Modeling Ontology (WSMO) or WSDL-S [7, 10]. A developer needs to have an in-depth understanding of the languages to develop, publish and/or consume a Semantic Web Service [7]. But, in reality this will be difficult to exercise as the languages are complex to understand and need significant effort to get familiar with [7]. In addition, converting the existing services to semantic Web services will consume greater effort and resource [7]. Therefore, there should be an alternative way of constructing Semantics for the existing Web Services and upcoming services that do not have semantic definitions.

The semantic Web services are published using semantic description and annotations. Currently existing ontology are mainly either too generic, describing the top-level concepts, or domain

specific. So, yet it is not possible to provide a common understandable and cross domain description of services [11, 12]. Understanding and determining the relationship and their dependency between services need further assessment and research as well [12].

Finally, the current developments do not enable applications to discover, match and consume Web services without pre-arranged human interventions [3]. There is no reliable and automated model to automatically discover Web services that can satisfy requirements, match the information requirement, execute and receive the responses.

This work is an effort to develop a model of automatic matchmaking bridging the limitations of the current semantic Web service. In this model: -

- Currently services are just semantically described and annotated without their relationship and dependency information with other services [12]. Accordingly, a Service similarity graph will be developed to draw the similarity level between published services. In doing so, one can easily explore available services based on capabilities, constraints and information requirements.

- The semantic similarity measures developed so far are generic lacking to be suitable for services due to the fact that services and properties are developer defined (artificial artifacts) while the generic cases work for real-world concepts. We will analyze and propose our own specialized similarity measures.

- The proposed model will automatically analyze and construct service relationship (Service-Net) linking published services based on the services similarity and dependency.

- The model will also have a Consumer which will have access to large pool of Consumer's data. During matching and negotiation, the consumer will have the flexibility of using any relevant resource in consuming the Publisher's service.

- The model will exploit the information available in the Service-Net to match service queries automatically to achieve the user's goal.

This work will develop a model with enhanced Service Broker and Consumer: -

- The Service Broker: - builds and maintains service similarity graph (Service-Net), understands and interprets semantically annotated service descriptions and service queries, and compute similarity among published services as well as make matching to consumers service requests

- The Consumer (Client): - which submits semantically annotated service queries to the Service Broker, analyzes candidate service offering and make negotiation with service providers. The Consumer will intelligently collect and access large set of linked Consumer's data to satisfy the service provider's information requirement.

1.4 Objectives

This thesis has the following general and specific objectives.

1.4.1 General Objective

The general objective of this research is to develop a general purpose efficient and scalable model for semantic Web service discovery and matching.

1.4.2 Specific Objectives

To achieve the general objective, the following specific objectives are designed;

- ✓ Study and analyze existing Semantic Web service discovery and matching models, languages and frameworks.
- ✓ Review literature on different implementations, researches and recent developments in the area.
- ✓ Study and use development tools.
- ✓ Develop service and domain Ontologies that specify services and service profile attributes, relationships and dependencies.
- ✓ Develop required algorithms and concepts.
- ✓ Develop a prototype and experiment it.

1.5 Scope and Limitations

The scope of this work is to develop a model for semantic Web service automatic matchmaking. Web service composition and interoperation are out of the scope of this work.

1.6 Methodology

In the process of achieving the objectives of this thesis, the following methods will be followed: -

- **Literature Review**

This work is an extension of previous developments in the area. Vast number of researches are being conducted and improvements are continuous. As part of understanding previous attempts, knowledge and latest developments in the area, an exhaustive literature review will be conducted. In this work, journals, books, different publications and scholarly articles will be used as input and critically reviewed.

- **Adopt or develop an experimental domain ontology**

Ontology is one of the key aspects of semantic Webs services. In this work domain ontology will be adopted or developed that matches the implemented prototype. A special service relationship and dependency description ontology will be developed to facilitate the definition of relationships and dependencies of services.

- **Develop the model**

This will be the major effort of the work. A model which will satisfy the promised quality attributes will be developed.

- **Experimentation**

In this method, the developed model will be experimented based on different criteria. A suitable evaluation approach will be proposed as well.

1.7 Thesis Organization

The rest of the thesis is organized in four chapters. Chapter 2 will present literature review to highlight the development in the area, Chapter 3 will also discuses selected works in matchmaking and related area of study, Chapter 4 will present our proposed model with its detailed analysis, algorithms and working formula, Chapter 5 will present our prototype and experiment result while in Chapter 6 we will provide conclusion and summarize the contribution of this work.

Chapter Two: Literature Review

2.1 Background

The World is going through dramatic and continuous growth of digital information available on the Web. Currently, every sector is significantly impacted by information technology - Governments, business and individuals use sophisticated information technology equipment and tools to perform their tasks. The wide spread of the technology attracted providers in both hardware and software manufacturing sector. In line with the increase of technologies and vendors, the demand of interoperability- information and service sharing and exchange becomes more demanding than ever [1]. Academic and industry researchers are striving to develop methods, tools and techniques to enable seamless integration of technologies across different platforms - vendors and technologies.

Sharing information and/or services across resources distributed over a network is one of the challenges of today's computation. Distributed computing is challenging due to the fact that it consists diverse and discrete software agents that need to work together to achieve some common goal [13]. These distributed software agents do not share memory, can be located in different sides of the world and can be operated on different technology platforms. As a result, computation across distributed environment is naturally slower and unreliable [13].

Computation in such distributed environment is highly challenging as there is no control and guarantee over the infrastructure and the behaviour of the agents. Different technologies and architectures are being proposed and used to overcome these challenges. Even though there is no indication as Web services of SOA has overcome distributed computation challenges than other approaches, Web Service has attracted significant attention both in academic and industry researches and enjoyed rapid acceptance by practitioners. SOA is a type of distributed system architecture characterized abstraction of services and message and description orientation. Services are described by machine processable description and characterised by platform neutrality, granularity and network orientation [1,4, 14].

Web services are now enabling the current business–to–business data exchange and service interaction over the existing Internet regardless of technology, vendor and platform variance [3, 4]. The next sections briefly discuss the current developments in the area.

2.2 Web Service

According to W3C, a Web service is "a software system designed to support interoperable machine to machine interaction over a network. It has a machine processable interface described using the standard Web Service Description Language (WSDL). Other systems interact with the Web service in a manner prescribed by its description using Simple Object Access Protocol (SOAP) messages, typically conveyed using HTTP with an Extensible Markup Language (XML) serialization in conjunction with other Web related standards." [1].

As it is outlined in the definition, Web services are software systems designed for distributed computing over the Internet which run over heterogeneous platforms. In its simplest definition a Service is a software module deployed on a network accessible platform by the Provider [1]. Web services are considered most appropriate when the application needs to operate over the Internet and reliability and speed shall not be guaranteed, when providers and consumers run on heterogeneous platform, when functionalities are required to be exposed for consumption by other interested consumers and when there is no requirement of managing deployments at consumer side [13, 14].

Web service operation has a lot of uncertainties especially in speed, reliability and availability. None of them can be guaranteed due to the inherent nature of distributed computation [13]. In contrary to this, Web services overcome the interoperability requirements across heterogeneous platforms and vendors and dispersed geographic locations [1]. This is achieved by using open and standard based set of technologies including XML, SOAP, WSDL and Universal Description, Discovery and Integration (UDDI) [1, 14, 15].

Web services extensively use XML for exchanging messages and defining services. XML provides neutrality, extensibility, readability and flexibility reducing the burden of inventing new technologies. SOAP also provides a standard, extensible, composable framework for packaging and exchanging XML messages over the Internet. WSDL provides a framework to describe Web services in a platform and vendor neutral form. WSDL enables a service provider to describe Web service's specification in machine processable format explaining input and output parameters and their type, associated constraints and other details of access. It also includes the location of the service provider – IP and Port addresses. UDDI acts as a central repository for services exposed by their providers [1, 15, 16].

2.3 Web Service Architecture

The Web service architecture consists of three components namely the Service Provider, the Consumer and the Registry [1, 5, 15].

Service Provider is the entity that exposes or advertises its service. In other terms, the Provider is the owner of the software module/component that renders the promised service. The provider, once implement its business services in its preferred technology and infrastructure, advertises its service to the rest of the world by publishing it on Service registry. The Provider shall describe its service adequately using standard WSDL to enable consumers find and use it. The Provider successfully hides the details of the Service's implementation by exposing only an abstract definition of the service. As long as the exposed abstract definition remains intact, the Provider has the freedom to change the implementation details without impacting its subscribed consumers [1, 14].

Service Consumer is an entity that uses services advertised by Service Provider/s. The Consumer with the help of the Service Registry discovers and uses the services published by providers to achieve some goal. The Service consumer is free of the implementation details and management overheads. It solely depends on the Provider to obtain the promised service as long as it meets the specified requirements [1, 14].

Service Registry is the central repository of advertised services in a searchable and discoverable format. It is the broker/intermediate between Service Providers and Service Consumers. Services that are available for consumers are advertised on the Registry using the machine processible WSDL. Service consumers search the registry for services that match their requirements. When a match is found, the registry informs the consumer the exact address of the service provider for further binding and invocation. The role of the Registry is until the Consumer obtains a matching Provider. During binding Consumer and Provider directly interact directly [1, 14]. Figure 2.1 [1] depicts the interaction among the three components.

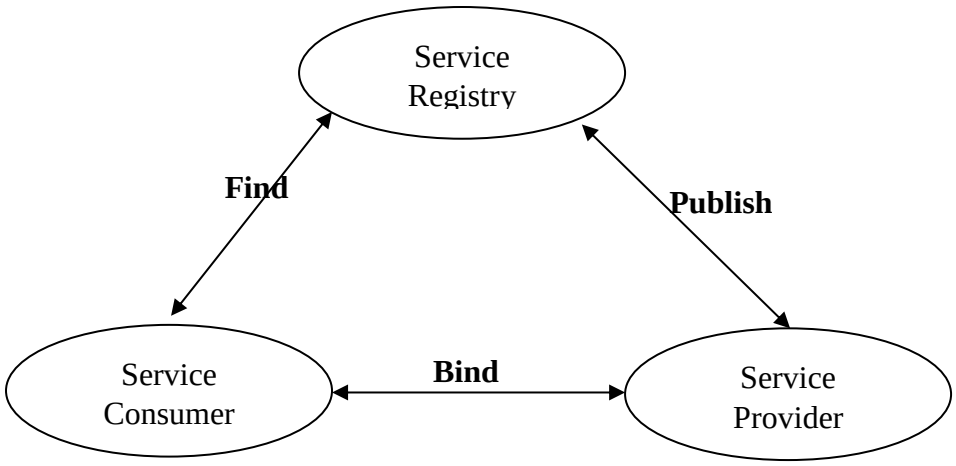

Figure 2.1: Web Service Architecture

Despite there are variants of the above architecture based on the business need and available infrastructure, the above architecture depicts the components and their respective role [1].

2.4 Web Service Engagement Process

Web service engagement process involves the three components of the architecture at different level. The engagement process encompasses three main activities – Publish, Find, and bind [1,5, 14].

The process starts when the Service provider publishes its service on the registry following the agreed standards of service description. After service advertisement, the Consumer shall know the existence of the service by some way. Usually, the Consumer searches the service registry to find a matching service that fulfills its requirements and meets its business need [5, 14]. The find operation happens twice. During design time the developer needs to find the service to understand and develop according to the specified interface description and during run time to retrieve the service's binding and location description [16]. During find operation the service Consumer needs to understand the interface description – the type of service, the data requirement and the result, and develop matching consumer accordingly. After implementation, during engagement the service Consumer initiates interaction with the Provider by providing the required data requirements of the interface [4]. It uses the binding details of the service to locate, contract and invoke the service accordingly.

In the current environment, most of the engagement processes are performed manually [3]. Though there are undergoing efforts to automate the engagement process fully or partially, it still requires significant human intervention [4, 16, 17].

2.5 Limitations of Web Services

The Web service technology enjoyed wide adoption and it is the most widely implemented among the distributed and interoperable technologies. The collaboration of WSDL, SOAP, UDDI and XML makes Web service a reality and a preferred choice. But these technologies are mainly keyword based and syntactical, significantly inefficient, hardly machine understandable [3, 5, 16] and vulnerable for security breaches [16] – especially for Denial Of Service (DOS) and Distributed Denial Of Service (DDOS) attacks.

As explained in the Web Service engagement process, the major limitation of Web services falls on the discovery process [1]. The web service standards like SOAP and WSDL are designed to provide description of messages and services for transportation and describing service interfaces. As a result, either of the two are helpful in locating services [25]. The intended/potential consumer shall know the existence of the service with its specification details, so that it develops a matching consumer agent. It is the responsibility of the Consumer to manually locate the service, understand the specification and develop a client that consumes the services [17]. This process is inefficient and hardly scalable. Useful services which are not promoted or noticed by consumers remain unused. Therefore, it is solely the Consumer's responsibility to locate, analyze and implement advertised services [3, 17].

For services implemented following the tedious searching and binding process, finding a replacement and making appropriate correction is also an overhead of the Consumer [3, 17]. It is also manual activity with the same process. Even if there are other thousands of service providers providing the same kind of service with exactly similar requirement, there is no way for the Web Service technology to be able to locate the other available services and automatically rebind to the working one when the already implemented binding fails [16, 17].

Similarly, as the binding is keyword based and direct, a slight change in the specification of the service will result in failure [3, 16, 17]. As consumers are widespread, communicating changes is not easy and feasible. Providers are restricted to maintain known specifications (method signatures

and constraints) intact to keep their subscribing systems working. So, cost of changing Web service specifications is high and complex. The semantic Web technology comes up to leverage some of the gaps discussed above through providing machine understandable descriptions [3, 4, 17]. The merger of semantic with Web service resulted in a Semantic Web Service technology that has brought about considerable improvement to the Web service technology and architecture.

2.6 Semantic Web

The Web has already acquired continuously increasing and enormous data. Due to its initial design and implementation, the Web is primarily for human readability and understandability [8]. Authors for the Web have been focused on providing information to human readers using markup languages like HTML. Through time the Web emerged as a very huge information repository that needs machine processing in support of human users [8].

The existence of such a huge and ever-increasing information on the Web demands for efficient information processing capability that cannot be achieved by human users. As a result, enabling machines to understand and process the Web and do useful activities on behalf of human users is the current direction of research works [9]. The Semantic Web has emerged as a dominant approach to achieve this phenomenon. As initially proposed by Tim Barnes-Lee *et al.* [9] the Semantic Web is the extension of the existing Web through explicit definition and standardization.

The Semantic Web promises a Web where humans and machines work cooperatively [8, 9]. A number of technologies and tools are developed in support of this – to transform the existing Web into a Web of data that can be understood by both humans and machines. These technologies enable to explicitly define the Web and build machine understandable knowledge. The Resource Description Language (RDF), the Web Ontology Language (OWL) and the Extensible Markup Language (XML) are anticipated to transform the Web into a huge Web-accessible database.

2.7 Semantic Web Service

Semantic Web service or semantic service in short is an extension of the traditional syntactic Web service through incorporating semantics. It provides machine interpretable information to Web services in enabling intelligent access to distributed services over heterogeneous platform [3, 4, 17]. Due to the syntactical nature, the conventional Web service definition and description lacks machine understandability hindering automatic discovery and matching [16]. Buying concept of

explicit annotation and description from the Semantic Web, the Web service emerged into Semantic Web Service through defining service specifications and service requests in machine understandable and processable way [3, 4, 8, 17]. The collaboration of the two emerging technologies enables for explicit description of a service capabilities and its inputs, outputs and constraints in a machine understandable and processable format while remaining interoperable and vendor neutral and accessible over the Web.

Semantic Web services are Web services in which published service descriptions are annotated using Ontology to enable agents representing service consumers to discover, interpret and invoke them autonomously [4, 8, 17].

As discussed in Section 2.5, the Web service technology comes up with different limitation that their semantic counterparts promised to address. Semantic Web services are expected to overcome the syntactic and keyword based natures of Web services. In Semantic Web services, discovery and matching processes are expected to be easy and automated. Semantic web service consumers will be able to discover and match services that can fulfill their requirement during run time and without prior arrangements of developers [3, 4, 17]. Unlike the traditional web services there will be no hardly encoded attachment between service consumer and provider, rather the discovery and matching processes will be performed at run time and automatically [8].

The issues of relocation or replacement of unavailable services are expected to be easily addressed as the discovery and matching processes are automated [3, 4, 17]. The Consumer and Service brokers have clear understanding and intelligence of the services provided by the Provider. As a result, the search is not mere keyword based but based on the actual capabilities of services. Unlike the traditional Web services there is no need of prior arrangement and communication between the Consumer and Provider developers. The Provider develops and publishes its services, properly annotating the capabilities and input, output and preconditions (IOPs), without prior knowledge of potential consumers. On the other side, the Consumer develops its consuming agent/function through semantically describing its functional requirements and without the knowledge of Provider's requirements or even existence of the service. The Consumer agent is expected to discover the Provider and then make matching arrangements without the involvement of a human user.

In Semantic Web service the service description, inputs, outputs, preconditions and constraints collectively called (IOPRs) are explicitly annotated using semantic ontology by the Provider [3, 4, 17]. In the same way, the Consumer is also expected to annotate its service requirements [3, 4, 17].

2.8 Semantic Web Service Architecture

Semantic Web Services Initiative Architecture Committee (SWSA) [6] proposes an architecture for the semantic Web service which extends the Web service architecture, in the aim to address the semantic agents' requirements of dynamic service discovery, service engagement, service process enactment and management, Semantic Web community support services and quality of service (QoS).

The architecture is built on the two emerging technological concepts of Web service and Semantic Web. It maintains the publication of service interfaces on the Web using WSDL, the invocation protocols like SOAP and HTTP and the Web published and accessible semantic description using dynamically linked and shared ontology from Semantic Web.

Like the Web Service architecture, the Semantic Web Service architecture also has the three stakeholders - the service provider, service consumer and matchmaker or service broker. Service providers publish their services to the matchmaker (broker) described using the shared Ontology and the consumer uses queries to interrogate the matchmaker for services matching its capability requirements. The entire architecture of Semantic Web Service works in the same fashion as their Web service equivalents, except the explicit description of services capabilities and service requests using Semantic Web languages. Figure 2.2[6] depicts the proposed architecture for the semantic Web service.

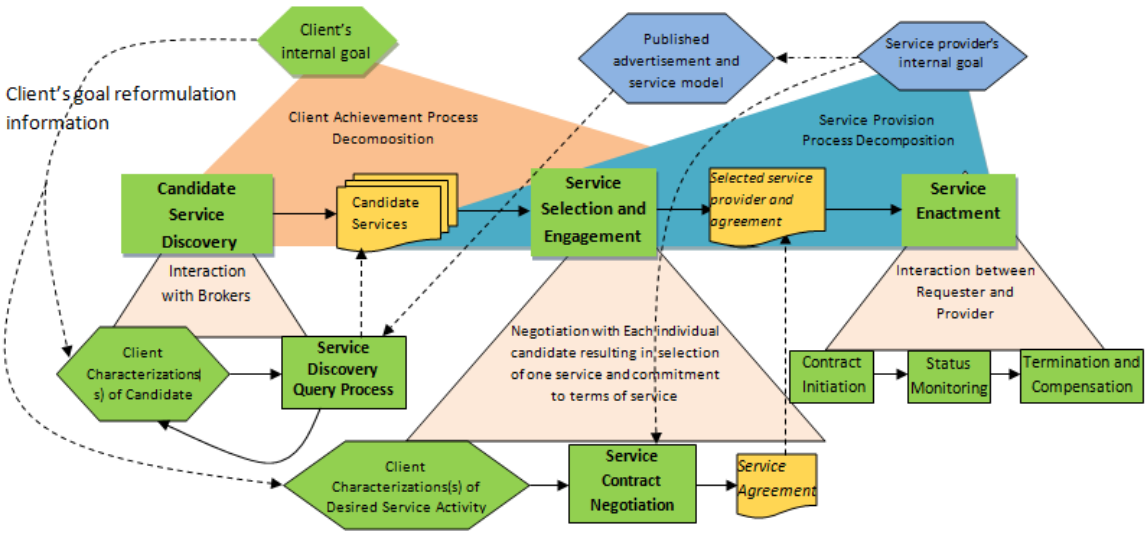

Figure 2.2 A Semantic Web Services Architecture

As shown in figure 2.2[6], the process of discovering and interacting with a semantic web services happens in three phases - Candidate Service Discovery, Service Engagement and Service Enactment. In the Candidate Service Discovery phase, a distributed search for available services that can meet the Consumer's requirement will be performed. The search could be performed through a centralized broker agent (matchmaker) or through a peer to peer interaction which share recommendations of potential services. During this phase the service providers indicate the services they provide, the service consumers seek services that fulfill their requirement and the matchmakers accept service description from providers and find matches to the consumers' requests. The service provider advertises its services using publishing protocols on matchmakers and consumers use the query protocol to ask for a match for their request. As mentioned above, the above process is purely manual in the current Web services.

The second phase is Service Engagement in which the Consumer interacts with potential service provider, validate and/or negotiate on the constraints and the terms of service delivery. Finally, in the Service Enactment phase the Consumer and Provider interact through messaging as per the terms of contract agreed during the engagement process to fulfill their mutual business objective.

The architecture proposed in [6] discusses the language requirements, negotiation needs and protocols of messaging during the three phases of discovery and interaction. In addition, as different communities may not share same ontology, translation and mapping of different Ontologies is also available at the Matchmaker level for better efficiency of matching services

described by ontology from different sources. The architecture also provides cross cutting services for Quality of Service, security, privacy and trust functions.

2.9 Semantic Web Service Ontology

At the core of Semantic Web is the explicit definition of shared knowledge through Ontology, the Semantic Web Service also needs Ontology to efficiently and clearly define services. The need of automatic service discovery and matchmaking requires a language that enables machines to understand what functionalities are provided by a service and what inputs are required and what outputs are expected on top of the different constraints that shall be fulfilled [3, 4, 17]. This requirement demands specialized domain Ontologies capable of describing the service functionalities and associated inputs, outputs, preconditions and results – collectively called IOPRs. A number of member proposals were submitted to W3C while Semantic Annotation for Web Service Description Language (SAWSDL) is finally made the recommendation.

Semantic Markup for Web Services (OWL-S) [18] is a W3C member submission Web Service Ontology from a coalition of academic and industry researchers. OWL-S aimed to enable users and software agents to discover, invoke, compose and monitor Web resources offerings especially services at high level of automation. The OWL-S is an ontology based on the OWL specification that provides information about a service in three essential areas. What the service provides for Users? How it can be consumed? And how it can interact? These are provided by the "Service Profile", "Service Model" and "Service Grounding" components respectively. Figure 2.3 [18] shows the OWL-S top level service ontology.

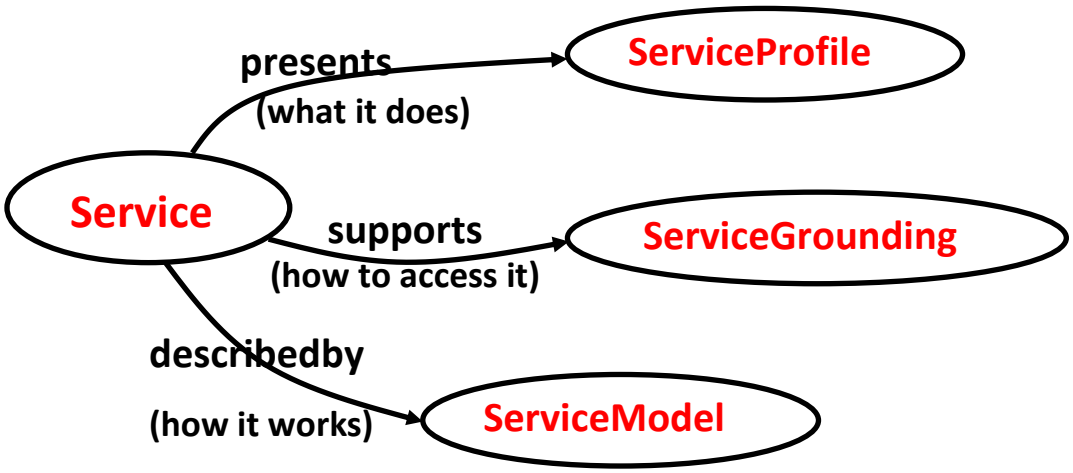

Figure 2.3: OWL-S Top Level Ontology

As per the specification [18] the Service Profile provides what the service does in a concise manner for service seekers. This includes what functionality the service provides, the restrictions and preconditions the service requester needs to meet including quality of service information. The profile ontology defines hasParameter, hasInput, hasOutput, hasPrecondition and hasResult properties to specify the Service's Parameters, Inputs, Outputs, Preconditions and Results respectively. The Service Model informs the Consumer how to use the service detailing the semantic content of requests and the different conditions and associated results. On the other side, the service grounding provides the details of access methods of the service. It typically specifies the communication protocol, message formats and other service specific details like IP address, port number and so on. OWL-S uses the WSDL and SOAP standards for grounding – actual execution of services on the current Web Service infrastructure.

The service profile provides utilities to semantically describe service parameters (inputs and outputs), conditions and results. It also provides textual description, naming and category information. It classifies services as atomic, simple and composite and provides the utilities to form composite processes by merging or composing of atomic processes using union, conditional flow, sequencing, split, choice or other forms of compositions. In general, the OWL-S ontology provides a domain ontology that specifies the artifacts of services and service descriptions.

OWL-S is the widely researched and discussed ontology with better support of software tools and arguably it is the near complete and neat approach that is built from the ground up objectively for the intended purpose [17]. The major shortcoming of OWL-S is its complexity lacking immediate integration with current standards of the Web service.

In a similar effort a group of researchers both academic and industry proposed Semantic Web Services Ontology (SWSO) [19] a description of the conceptual model underlying the ontology, and a description of a first-order logic (FOL) axiomatization which defines the model-theoretic semantics of the ontology. The axiomatization is called FLOWS -- the First-order Logic Ontology for Web Services. FLOWS is an effort to extend the OWL-S [18] by including missing capabilities that were not part of the initial objective.

Adopted from OWL-S, FLOWS has three major components – Service Profile, Service Model and Service Grounding. Service Descriptors provide basic information about a service including non-functional meta-information. SWSO recommended service description properties derived

primarily from OWL-S complementing the missing once from WSMO. As noted, it can be generalized that SWSO is a more detailed, elaborated and complimented version of OWL-S supplied by First-Order-Logic reasoning. SWSO discusses in considerable detail service types and composition ontology. Like OWL-S, SWSO is just a member submission to W3C with no official approval and has very limited near to non support of tools and research works.

Semantic Annotations for WSDL and XML Schema (SAWSDL) [20] specifies the methods of adding semantics to WSDL parts like inputs, outputs, interfaces and operations. Unlike the other proposals discussed above, SAWSDL does not introduce a new language or top level ontology to describe services rather it provides the mechanisms to attach semantic information to the existing WSDL. SAWSDL is made a W3C recommendation since August, 2017. The major building block of SAWSDL design is maintaining the existing extensibility of the WSDL and using the semantic annotations of services after the discovery process for negotiation and invocation processes.

The SAWSDL also defines mechanisms to map XML schema types to ontology concepts. This extension is also proven working to WSDL 2.0 and 1.1 as well as the XML Schema extensibility framework.

Out of the different WSDL components that represent service descriptions, SAWSDL focuses on semantically annotating the abstract definition of a service (Element declaration, type definition, and interface) that enable dynamic service discovery, composition and invocation. It provided a generic reference mechanism that can be applied for the above components.

It provides an extension attribute called **modelReference** to define the relationship between WSDL or XML schema component and a concept in the semantic model. The modelReference can be used to annotate XML Schema type definitions, element declarations, WSDL interfaces, operations and faults. ***liftingSchemaMapping*** and ***lowerSchemaMapping*** extensions are also added to XML Schema element declaration and type definitions for specifying mappings between semantic data and XML.

The major limitation of SAWSDL is that it does not provide a formal semantics to define service attributes rather it just provides a syntactical extension to WSDL. Though SAWSDL is adopted as a W3C recommendation in 2007, its acceptance in the industry is limited. As a result, tools and researches are significantly limited [9]. Despite the mentioned limitations, as it is the W3C

recommended standard and presumably a future direction in Semantic Web Service, our model is based on this ontology and framework.

2.10 Semantic Service Matchmaking

As defined in [5, 21, 22], service matchmaking or discovery is the process of finding a service, out of the available many, that best addresses the service Consumer's request matching functional and non-functional requirements. In both the semantic and syntactic web services, discovery is a critical process. Services are useless unless they are discovered and used by consumers [21]. As defined in [1] matchmaking is "the act of locating a machine-processable description of a Web service-related resource that may have been previously unknown and that meets certain functional criteria".

The discovery process becomes more and more complicated and challenging as the number of services are increased [22]. For any service request there will be many candidate services that promise to deliver the required service and matching the one that best suits the request is still a challenge and requires future improvements [21, 22]. The service discovery process is affected by the factors like the service provider's ability to describe its service, the requester's ability to formulate its requirement and the matchmaker's intelligence in finding a match for a request [5].

The current approaches of service discovery can be classified as either logic-based, non-logic-based or hybrid based on the nature of reasoning used for matching [23, 24]. The non-logic-based approaches use methods of graph matching, schema matching, data mining and text similarity measurements to measure the degree of match. The logic-based approaches perform logical reasoning on service descriptions and the hybrid approaches combine the above two approaches [24].

The logic-based approaches took advantage of the machine processable service descriptions and standard logic inferences [23, 24]. In these approaches the matchmaker computes the semantic similarity of service descriptions to service requests. The most common logic-based approach is a plugin measure in which the matching level of input, output, precondition and effects (IOPEs) provided by the service Provider's description compared against the Consumer's service request query in the scale of exact, subsumption, plugin and fail as initially proposed by Paolucci *et al* [25].

On the other hand, the service matching can be performed at service profile or process model level. The service profile matching commonly known as the "black-box" matching determines the semantic similarity of services based on the description of their profiles. The profile of a service describes what the service actually does in terms of service signature, input and output (IO), pre-conditions and effects (PE) as well as the additional non-functional aspects as provided by the service profile description. The service process-oriented matching also called the "glass-box" matching determines the level of matching of services based on the desired operational behaviour of services in their process control and data flow [23].

There are a number of matchmaker of semantic web services working on the different service ontologies –OWL-S, WSMO or SAWSDL/WSDL-S. The notable SAWSDL/WSDL-S matchmaker includes SAWSDL-iMatcher [22], SAWSDL-MX2 [26], URBE [27] and TOMACO [28]. Most of the matchmakers mentioned above are hybrid matchmakers that employ both logic-based and non-logic based matching to enhance the efficiency of their algorithm.

Service matching is the most important process in Web service technology [21, 25]. Services that are not discovered cannot be used as the same time the matching is a key process for service replacement. Consumers will ask for service replacement when the one they use or know fails or went unavailable. Therefore, finding some other service that can render similar service with the previous one is a matching phenomenon.

As it can be easily inferred from previous works [22, 26, 27, 28] and assessments [5, 23], the matching process and algorithms yet need improvements and stability. The existing methods are not sufficient enough to address the aspired objectives of automatic service matchmaking in particular and the Semantic Web in general.

2.11 Semantic Similarity Measure

Semantic similarity of concepts is the measure of the similarity degree of two concepts in a given ontology. There have been different proposed approaches and algorithms to measure the similarity of two concepts. The existing similarity measure approaches can be categorized as edge counting or Knowledge-based, information content and hybrid approaches based on the techniques used [29]. Edge counting measures depend on the shortest distance between the two concepts, exploiting the hierarchical taxonomic information following the "IS-A" relation [30], while information

content measures the probability of the concepts occurrence in a corpus. The main factors in measuring semantic similarity in ontology, as per the edge counting approach, are the path length, depth and local density [29, 30]. Edge counting approach computes the distance of the path linking the two concepts. There are a number of variants to this approach.

The work in [31] introduces the path-based approach measuring the shortest path between two concepts. The similarity of two concepts is measured in comparison of the maximum distance against the shortest distance of the two concepts in the taxonomy. (Sim (c1,c2)=2*Max(c1,c2)-SP) where Sim (c1,c2) is the measure of similarity between concept 1 and 2, Max(c1,c2) is the maximum distance between the two concepts and SP is the shortest one. Wu and Palmer [32] extended the work in [31] through incorporating the depth of the Least Common Subsummer (LCS), defining LCS as the most specific concept the two concepts have as an ancestor. Furthermore, Leacock and Chodotow [33] further extended the works in [31, 32] by adding the maximum depth of the taxonomy in the computation. This work offsets the path distance against the maximum depth of the ontology resulting in improved similarity ranking. The proposed work in [33] computes the similarity of two concepts as a negative logarithm of the counting the shortest path between the two concepts divided by twice of the maximum distance from the shared concept (Least Common Ancestor).

The evaluation of the algorithms as reported in [29] yields a close performance of the path based approaches with the work reported by Leacock and Chodotow [33] showing slight upper hand over the others.

2.12 Negotiation

Negotiation is the process of reaching an agreement on the terms of cooperation and collaboration between two or more actors that have some common interest but conflicting preference [34]. The objective of each actor is to maximize its respective benefit while protecting its internal interests. In cases where the negotiating actors have conflicting interest, the negotiation will be tough as each actor is attempting to exploit the weakness of the other actor to maximize its benefits. On the other side, when the interest of the actors is not conflicting rather cooperative, each actor contributes its best cooperatively to achieve its common goal.

In recent years, the development in Negotiating Software Agents (NSAs) mainly focuses on marketplaces and personal assistants in which agents autonomously or semi-autonomously negotiate to purchase or set personal arrangements on behalf of human users. The negotiation works for NSAs are full stacks including messaging protocols and frameworks from initiation to decision making.

Negotiation is a complex task and automated negotiation is more complex [34]. Determining the negotiation strategy, negation protocol and developing negotiation ontology are the key factors of the complexity. While the open negotiation approach is more complex as the negotiating actors and negotiation agendas are all open, the controlled one provides a controlled negotiation environment in which certain areas of the negotiation are restricted.

2.13 Summary

Web services are primarily celebrated for their modularity, simplicity and neutrality that enable vendors with different technology and platform preferences to easily exchange information and integrate services over the existing Internet. But not that long when the mere syntactical nature of the technologies around Web services became a hindrance for further improvement. Semantic Web services or semantic services in short are the next generation of Web services that promise to address the major shortcoming of web services.

The Web services shortcomings, especially their lack of machine understandability, demand developers to manually match services at design time and make run time service replacements unthinkable. With the advent of semantic web, Web services collaborate with it to address this major limitation promising for dynamic service matching and easy discovery. Leveraging developers from making all the tedious service matching process automated.

In an effort to realize it, developments are made in developing ontology for service description SAWSDL/WSDL-S [20] being a W3C recommendation and a number of matchmaking algorithms are proposed like the SAWSD-MX2 [26]. Despite the aspirations and the theoretical possibilities, the development so far has not made it a reality. The semantic service technology still needs significant improvement and maturity to deliver its promises. Ontology for service description, algorithms for matchmaking and a proven model are required.

Chapter Three: Related work

In Chapter two, we have assessed Semantic Web, Web Service and Semantic Web Service to introduce the general developments, concepts and theories. We also identified the gaps of current developments specifically in achieving automatic service discovery and matching. In this Chapter we will review related works that attempt to address similar limitations.

It is clear that Web services are useless unless they are discovered [5, 12]. Discovery is the process of finding the suitable service that fulfills the consumer's functional and non-functional requirements [12, 40, 49]. This definition has attached quality attributes of time and cost.

As discussed in [40] the service discovery and matching process is affected by the factors of service and request description and formulation and the matchmaker's ability to match the request to the existing published services. The two factors go hand in hand as the intelligence of the matchmaker is significantly dependent on the abilities of the two actors (Provider and Consumer) to describe their capability and requirement. The need of automatic service discovery and matching requires a language that enables machines to understand what functionalities are provided by a service and what inputs are required and what outputs are expected on top of the different constraints that shall be fulfilled [5, 37, 41].

Regardless of the architecture of the discovery, broker based or peer-to-peer, the description is a dominant factor followed by the algorithms making the matching. In the following sections we review notable works of matchmaking, service description languages and negotiation.

3.1 Matchmaking

Matchmaking is a very important aspect in the web service engagement process. As stated in [7], it is the process of locating a machine-processable service description that satisfies the Consumer's requirement and which was not known previously. The major limitations of the traditional Web services matchmaking were the need of human engagement in the process. The semantic services are expected to ease the process making automatic matchmaking a reality. The matchmaking algorithms, proposed so far, are categorized as logic-based, non-logic based and hybrid based on the nature of reasoning employed [51].

Adala *et al.* [7] proposed a framework that uses Natural Language Processing (NLP) to process Web service descriptions and user queries expressed in both formal and natural languages. The authors argue that due to the lack of shared common ontology, the limited popularity of service ontology languages and the skill to utilizing them and the already existing Web Services matchmakings based on natural language is preferable. They also claim that as there are a number of ontology languages in use to describe services and most of the matchmakers are specific to languages, their approach has the neutrality of working with all service ontology languages including natural languages.

According to [7] keyword based approach has advantages over the semantic based approach due to its simplicity, familiarity to users and its open vocabulary. Unlike formal languages which demand users to have expertise in semantic languages to formally specify services, the keyword approach does not require users to know anything more than their natural human understandable language. This also has an added advantage as the need of sharing ontology or employing the ontology mapping service is not required as it is already expressed in terms of natural language which is common across the actors. It is noted that keyword based approach is widely used in other matchmaking activities like in search engines.

In this proposed framework, there were no assumptions taken in terms of description languages. It opens the use of natural languages or other semantic Web languages. Web services could be described in WSDL or semantic languages like OWL-S [45], SWSO [47] or SAWSDL [48]. In this approach the most important questions were how to extract the most relevant and representative keywords form the description, how to map the user queries to service descriptions and How to map natural language keywords to Semantic concepts?

In addressing the above questions, the authors used Natural Language Processing techniques and tools to extract senses from keywords submitted by users and from Web Service descriptions. The model also contains mapping module which converts English terms from WordNet [35] to Suggested Upper Merged Ontology (SUMO) [36].

Web Services described in Web WSDL, OWL-S or WSMO and published at service registries are parsed to extract useful information. Natural Language Processing (NLP) techniques are then applied to get the most useful words for next steps. Word disambiguation is also used to

disambiguate words to be followed by WordNet to SUMO mapping. Semantic matchmaking is then carried out based on the distance between concepts.

In the service Consumers side, the framework provides a simple user interface to submit user queries in natural language. Keywords are extracted from the user query and pass through same processing as the Service description – Natural language processing, WordNet to SUMO mapping and semantic matching. The processed user query will then be matched against the already processed and stored service profiles.

Though this work discussed and tried to address the outstanding problems of semantic Web services, the proposed approach did not address the problems sufficiently. The proposed solution is not different from the existing keyword based Web Service matchmakers that demand significant human interference. It did not provide semantics to Web services as it has initially promised and analyzed. The final result of this work presents a user interface to enable users search published web services based on keywords. Accordingly, automatic Web service matching is not achieved. Still after searching the services the user needs to determine which service to use and needs to manually develop the Consumer.

Rathore and Suman [37] experiment a model for dynamic discovery and composition of Web service using Quality of Service (QoS) computation. The work aims to address the existing gaps in Web service discovery and composition. As the number of Web services providing similar or same service are increasing, choosing one out of these available services requires careful analysis. Based on the QoS approach the quality of service is computed to discover the most appropriate and efficient service out of existing similar services that match the user's request. As this computation is performed at run time and due to the existence of many similar services that are candidates to satisfy the users request, this activity is usually time taking and uneconomical. Accordingly, in this work the authors proposed a broker-based approach to compute and maintain the QoS and reputation of services.

They claimed that the approach significantly improved the service discovery and composition process using Local Selection and Local Optimization (LSLO). It is also better in fault tolerance and ensured improved reputation predication using Mixed Opinion based Reputation Prediction (MORP).

In this work, the authors incorporated possible improvements and functionalities in the model and claimed success in their work. But the work is syntactic based and it is obvious that the dynamic discovery and composition is only dependent on QoS. The usual concern in QoS based approaches is specification of quality and storage of quality information [7]. They haven't discussed how services similarities can be determined or how candidate Web services that can fulfill the users request can be selected. Moreover, new services will have serious difficulty to be discovered and noticed due to lack of reputation and recommendation.

Benaboud *et al.* [38] proposed a Semantic Web Service discovery framework using intelligent agents and ontology. Though they acknowledged the role of Ontology in Web Services for improved machines understandability, they stated that the Ontology by its own did not enable machines to be intelligent enough and act by their own. Accordingly, the authors proposed intelligent Agents to augment Ontology and enable computers understand and make decision in the Web service discovery process.

In this approach, team of agents analyze their local information and cooperate in the discovery process. This is especially exploiting the intelligence of agents in making decision at changing environment and available information which the traditional systems are not capable of. QoS is used to find a better match when the discovery engine returned more than one service provider for the consumer's request. Consumers also have the privilege to weight quality attributes according to their preference. The model uses the OWL-S [45] ontology language to describe functional requirements recommending their own extension to add non-functional description. The functional requirement is about the service profile, service name, textual description, input and output, while the non-functional is about the Consumer's preferences mainly in terms of QoS.

The Consumer agent is responsible to forward its query, build using the provided Ontology-guided user interface to select the desired terms from the list of terms provided, to potential service provider agents. The service provider agents match the Consumer's query to service profiles stored in their registry and submit the candidate services to the Consumer agent for further analysis based on QoS attributes and other matching criteria implemented by the Consumer itself.

As mentioned above, the model computes similarity of user queries to service profiles in a two step process, at the Providers and Consumer side, the provider computes the functional match

while the Consumer does the QoS. Functional match, between the Consumer's query and the services available in the registry, is performed using both syntactic and semantic matching approaches. The syntactic similarity of service name and description is performed using string similarity matching techniques while semantic similarity of inputs and outputs is computed using the plug-in matching similar to the work in [7].

Though the model attempts to avoid the middleman, the Broker agent, it has communication inefficiency as the Consumer will not possibly know the potential service providers. As a result, the Consumer agent will be forced to broadcast its request to all known providers, which would congest the traffic and also the Providers consume significant resources to analyze every coming request. On the other side, the primary matching technique is syntactical that would not address the known limitations of false positives and negatives.

A work by Paolucci *et al.* [25] focusing on locating Web Services on the basis of the capabilities they provide is a prominent work which is considered a pioneer in semantic Web service matching. This work has identified the need of a language to express the capabilities of services and the specification of matching algorithm to match between advertised services to requested services. The authors adopt the DAML-S, later called OWL-S [45], as a service description language.

The proposed method extends the Web service architecture adding a semantic layer. This was due to the limitations of the Web service specification to locate services based on their capabilities. The UDDI of Web service is augmented by set of attributes called TModels, which describes additional features of services that cannot be maintained in UDDI. The limitation of XML has also imposed significant challenge as two identical XML descriptions may mean different things depending on the context of use. As a result, capability based matching will not be achieved without adding a semantic layer.

In this work the authors proposed Web wide registries of Web service that can act as directories for advertised services. These directories act as central registries for advertised services with matching capability. Their matching algorithm is based on *sufficient similarity* between the advertised and requested services. Sufficiently similar is a level of similarity between advertised and requested services in a range of exactly similar to completely dissimilar. It can be exact match or render some level of relatedness. Similarity degrees are rated as exact, plug in, subsume or fail

in order of strength from the strong to the weakest. The matching algorithm considers a service as matched to a request only when the service can render some value to the Consumer. More specifically, a service is considered as a match when its inputs and outputs exactly match to the Consumer's counterparts. This criterion is introduced to guarantee that the matched service satisfies the Consumer's request at the same time the Consumer also supplied the candidate service sufficient information to operate.

In above work [25], a match is considered if and only if for every output of the request there is a matching output from the advertised services and for every input of the request there is a matching input on the advertised side. If any of the above conditions fails the match is also considered failed. The match between the inputs and outputs is based on the relations between the associated concepts and the degree of match between the concepts is computed by the minimal distance between the concepts in the taxonomy tree. They specify four levels of matching degrees, exact, plug in, subsumes or fail, commonly referred as plug-in or subsumssion hierarchy.

- Exact: - is when a direct match is found from the advertised to the request or when the request is a subclass of the advertised one,
- Plug in: - is when the output of the advertised subsumes the output of the request (Request is subclass of Advertised),
- Subsume: - is when the output of the request subsumes the output of the advertised (Advertised is subclass of Request), and
- Fail: - is when there is no any kind of matching between the two.

Exact is a strong match that fulfills the requester's need followed by plug in as it can render same or equivalent level of service while subsume may only partially fulfill the request's need.

The above matching model is one of the pioneers and the most referenced work in the area of semantic Web service matching. This work, like other works in the area, considers matching of web Services using Outputs and Inputs as primary factors. While the subsume approach remains the dominant in the area of semantic web service matching, this work expects the Consumer to know every input and output of the operation in advance and the matching is based on direct match of inputs and outputs of the two counterparts. This hinders the Consumer from making dynamic matching as it is not practically possible for the Consumer to know the required inputs in advance.

The matching levels are the four mentioned once (Exact, plug in, subsume or fail) without any quantification. Additionally, the computation is not clear when there are more than one inputs and outputs.

Another work by Samper et al. [39] proposed a semantic Web service matching algorithm exploiting the additional category information offered by the service profile in OWL-S. In this proposal, all published services that are in the same category with the Consumer's requested services will be traversed and tested for matches with the client's requested services. Then the parameters of the returned lists of services will be compared with the request's parameters and different degree of similarly will be assigned. At the end, a composite calculation will be made considering the input, output and capabilities rating. The one with better score will be matched for the request.

In this work the authors extended an algorithm for service capability description which was initially proposed by Paolucci *et al*. [25]. The authors claimed improved performance of their work in comparison with previous attempts. The approach is obviously inefficient as it evaluates all services under the same category to determine their similarity level against the request service, comparing parameters and capabilities, every time a request is made.

In a similar effort Benaboud *et al.* [40] proposed a model which uses Agents and OWL-S assisted by user preferences. In this model, recognizing the inefficiency of traditional Web service discovery and matching approach, especially due to its keyword dependency and high human intervention, they proposed a model which uses software agents and OWL-S.

This model has four layers. The first one is Web service and request description layer which acts as a registry of services and entry channel for service requests and uses OWL-S to describe both services and requests. In this layer publishers describe both the functional and non functional capabilities of their services using OWL-S while consumers also submit their request describing in the same way. The functional match layer is the second layer which computes both syntactic and semantic similarities between published services and requests. The third layer is QoS layer which computes and compares the service quality of published services when layer two returns more than one match for the request. The fourth one is reputation layer; at this layer the reputation of services is calculated.

If more than one service is returned after the fourth layer, a composite calculation will be made to select one service which has the highest score. This model provides users to describe their preferences and also gives rating for their satisfaction – reputation.

Pakari *et al.* [41] experiment a novel approach by combining the known matchmaking approaches of syntactic, semantic and structural similarities matching. In this work, after analyzing the advantages and disadvantages of existing matchmaking algorithms – syntactic, semantic and structural similarities, they proposed a hybrid approach utilizing the advantages from the three methods.

They compute the syntactical similarity between the search query and the service description using Jaro-Winkler's strategy. According to the strategy, it computes the number of matching characters in the request and description strings and the number of transpositions. After computing syntactical similarity, they use WordNet to evaluate the structural similarity between the request and description concepts. This enables to identify the similarity of the two concepts based on the path length between the WordNet taxonomies. Finally, they use domain ontology to assess the similarity of the concepts. They drive the relation between the concepts based on the ontology. It uses "subclass of", "Superclass of", "equivalent" and "disjoint" notation in determining the similarity of concepts.

After computing the three similarities, they computed a weighted average to reach the final result – by giving more weights to the semantic similarity while the two others are weighted equal. In this computation, they consider the Input, Output, Result, Preconditions, Service Name and Service Descriptions.

The result of this hybrid approach is compared against other works which used one of the approaches, and the authors claimed obtaining better result in matching services for a request. The major problem of this work is its inefficiency. It has to compute three layers of similarity to reach on the final decision and the structural and semantics similarity computations will consume much of the effort that make it significantly inefficient. Secondly, the value addition of syntactical and structural analysis is not obvious. The result is mainly dependent on the semantics similarity. In addition, they haven't assessed the possibility of obtaining more than one services satisfying the

criteria. There is no clear indication what their approach will do in case where more than one services have similar result of the computation.

In similar effort Fenza *et al.* [42] proposed a hybrid model for Semitic Web Service matchmaking which uses a fuzzy matchmaking and clustering algorithm for grouping OWL-S documents. In this model, the central role is given to task-oriented agents that discover approximate match for service requests when there is no exact match. The model uses fuzzy multiset mathematical model to represent the multi-granular information of OWL-S based descriptions in Semantic Web Services.

This model is build based on two distinct layers – the knowledge and agent layers. The knowledge layer is acquainted with all the knowledge processed and maintained by the system. Ontologies, taxonomies, information processed by the agents and stored by the system are the building blocks of this layer. The agent layer is process oriented and classified into two as advertiser and broker agent. The two agents represent the Service Provider and Service Consumer sides respectively. They manage the interaction between the two components by filtering and interpreting the Provider's service capabilities and the Requester's queries. They interact natively to perform tasks in collaboration.

The Service provider's agent interfaces between the Provider and the Broker agent. It translates the relative OWL-S service specifications into a concept based representation by means of fuzzy multiset model. The Broker agent performs brokerage and matchmaking between the Consumer and the Provider. It mediates between the Provider's and the Consumer's agents. It interprets the Consumer's query and then interrogates the Service Provider's agent to find the Web services that best fits the Consumer's request. This approach uses the well known Fuzzy C-Means (FCM) algorithm [43] for clustering.

The major issue of this work is efficiency and low matching. As it can be seen clearly, the Broker agent is over burdened handling nearly all activities of the process – from discovery, matching, clustering and mediating. It is the central and the major player which orchestrates all activities. This has a clear issue of efficiency. On the other hand, the model promised and designed in the way to return an approximate match when there is no exact match. Therefore, the model will return

a match whenever a request is submitted even if the match is significantly low and even cannot fulfill the Consumer's need.

Heidary *et al*. [44] proposed a three phase matching model. This model performs matching of Semantic Web Services on the basis of input and output description. It takes the advantages from a graph and flow networks. The approach assigns matching scores to the semantically described inputs, outputs and their types. It makes a flow network in which the weights of the edges are the scores using the Ford-Fulkerson algorithm. This model demanded all services to use same ontology for description. Among matched candidate services the one with the least running time will be chosen.

In this proposed model, at its first phase, it semantically compares the parameters of two Web services. It also compares the capabilities and functionalities of these two services. In its second phase, it computes the similarity of the types of parameters. Finally, it computes the matching rate of results in phases 1 and 2. The main objective of this work is to find some other service that has similar method signature so that when the one in use fails, the one with similar signature will be replaced.

This work has not considered the existence of services with exact method signature but different functionality. The major emphasis of the work is on computing the similarity level of the services inputs, outputs and their types. In reality, services may have similar signature but completely different capabilities and constraints.

Chouiref *et al.* [45] also experiment a model for Semantic Web Service Discovery using similarity of contextual profile. The authors define contextual profile as a set of features the client has, like the client's terminal information, preferences, location, time, identity, etc. As a solution to the well known challenge of returning less relevant or not relevant matches to user queries by search engines, the authors propose a model which considers contextual profile of the client. The main idea behind this proposal is to understand the user's information to better understand the search query.

So far, the UDDI has no specification to specify the Consumers' contextual profile. Therefore, searching UDDIs have no filtering based on the Consumer's information. In this model, the authors

proposed to augment the existing UDDI with contextual information so that it will have better understanding of the Consumer's query.

The contextual information gathers and classifies static, temporary and dynamic information of the Consumer. Static data like personal data and dynamic data like preferences and temporal data like location and time are gathered and kept in the form of a tree that shows hierarchy of concepts. In the proposed discovery architecture, the authors proposed an interoperability module that has an administrator of the profile, inter-ontology similarity module and a contextual database and a Discovery and Selection engine which contains a contextual profile filtering module and similarity measure module.

To measure the similarity of Consumer requests and Provider descriptions, the authors improve the similarity measure defined in [46] by combining syntactic and semantic similarity measures in both qualitative and quantitative measures.

This work has discussed the proposed model theoretically and there is no implemented prototype and improvement comparison against any previously implemented models. The model is mainly to enhance search accuracy for human users but not for automated search by software agents.

In another work by Zohali and Zamanifar [47], a two phase approach is proposed using semantic similarity matching and qualitative filtering to improve the performance of Semantic Web Service discovery. This model proposed to extend the existing Web Service architecture by adding semantic information to UDDI.

The service matching model requires all services to be defined and published using semantic ontology language and to extend the UDDI to store semantic information. In the matching process input and output parameters are evaluated in addition to a user defined matching approach. The matching Web service is then selected using qualitative filtering as per the above evaluation.

The authors used the Semantic Web Service Description (SWSD) Ontology to semantically describe primary information (parameters and capabilities) and Quality of Web Service Description Ontology (QoWSO) to describe quality attributes. Semantic and Quality information of Web Service is stored along the UDDI in TModels [25] that describes additional features of services.

In this model a function based semantic similarity matching is used in which the input and output of a published service are evaluated against the request. When the request's inputs and outputs are subsets of the published counterparts, the matching process is considered successful. The user has also a chance to weight parameters and determine the most prominent out of the parameters. In case exact match is not possible, degree of matching is assigned in terms of sub-concept, super concept and fail for subsequent average composite score calculation. When more than one candidate services are identified, sorting is performed weighting those with high composite results.

The model uses QoS attributes of services in terms of stability, response time, reliability, access time and grade. The model added a Search Agent and QoWS Certificate Authority apart from the three components of the Web Service architecture. The Search Agent processes the interaction between Service Consumer and the UDDI. It parses the SWSD and QoWSO descriptions and performs the matching process as per the discussed approach. The QoWS Certificate Authority provides quality certificate for published services.

This work attempted to extend the existing Web Service architecture through adding semantic description and QoS attributes. The work expects the Consumer to request Providers with exact set of Input and Output parameters at hand. This work excludes the chance of negotiation between the two parties.

Similarly Dong *et al*. [48] experiment a Web service search approach based on service similarity. Unlike [25, 47], which experiment finding matching Web services based on the input/output parameters, this work has recognized the inefficiency of input/outputs and description for matching purpose. Neither the input/output parameters nor the descriptions are sufficient enough to determine the capabilities of services.

The authors proposed a technique that clusters parameter names into semantically meaningful concepts. The approach also considers both the description of the service and its parameters during analysis of similarity. Understanding the complexity of naming of parameters, they consider the co-occurrence of parameters in inputs and outputs to cluster terms into meaningful concepts. They refine the agglomerative clustering algorithm to suit the case. According to this work, similarity of service is based on the input/output and description similarity.

The works in [25, 41, 47, 48] expect Consumers to request the UDDIs supplying all required inputs which is not the case in reality. Service consumers will not know what is available and what is required during their search of services. As a result, the major expected flexibility is to negotiate on the inputs/outputs and determine them at runtime. In addition, matching on the basis of input and output parameters is significantly inefficient and unpredictable as these parameters are defined by the whim [48] of the Provider.

3.2 Web Service Similarity and Relatedness

On the Web, services serving same or similar functionalities are provided by several service Providers. Most of the time, services from one provider can be replaced by another provider's or can serve partial of it. On the other hand, Services on the Web are not isolated or self contained. In most cases services may depend on other services to provide their promised functionality. As the Web already presents a huge number of Web services, analyzing and understanding the relatedness and dependence of one service with/on another is crucial.

Service relationship analysis has significant importance in the discovery and matching process of web services. Services have relationships in terms of similarity and dependency that has importance in the process of discovering and matching. Service similarity is the measure of relatedness of Services' functionality. The level of similarity ranges from exactly the same to completely disjoint. Exactly same services function the same way, partially matched services provide the partial solution of a service provided by another while others are completely disjoint. In most cases, we may find several hundred providers for a typical service while the implementation details and solution approach may vary from Provider to Provider.

On the other side, a service may depend on another service/s to deliver its promised functionality. The dependency varies from being slightly/partially dependent on another service, the primary service performs the main functionality, to orchestration or composition, the role of the primary service is fully dependent on the child services. We assess some of the major works conducted to analyze the dependency of a service on another in the following pages.

Cherifi *et al*. [12] worked on a Web service dependency analysis with the objective of understanding services to improve discovery and composition. The authors perform analysis for both syntactic and semantic descriptions of Web services in perspective of complex networks.

They extract publicly available syntactic and semantic Web service descriptions and analyze using tools from complex networks to understand topological properties of services. They demonstrated the resemblance of Web service dependency networks to real world scenarios like community structures.

They constructed dependency network by connecting operations to their input and output parameters. Each operation and parameter represented by nodes, a directed edge is drawn from the input to the operation while a reverse directed edge is drawn to outputs. The matching function works based on the level of similarity of parameters.

Similarly Gu *et al.* [49] experiment an enhanced service dependency graph approach for service composition. Their approach uses static declaration of dependencies to construct the service dependency graph. The original specification of WSDL does not explicitly allocate attribute that can maintain service dependency information. But as it is flexible and accommodative to certain extent, the authors extend WSDL to integrate dependency information.

The work in [50] has also exploited a similar approach in the aim of service composition. In this work automatic extraction of dependencies is investigated using semantic similarities of input/output parameters, constraints and conditions.

As demonstrated in [12, 49] connecting services based on their dependency on other services has improved the discovery process. While automatically analyzing the dependency of one service over another is still a research area and considering the input/output parameters as dependency measure has serious limitations, it is workable to use a manual approach as demonstrated in [49]. Web service developers can supply additional information that indicates the Service's dependency in addition to the known attributes.

3.3 Service Ontology

The core idea behind semantic Web service is the association of formal meaning to the different parts of service description. Different proposals have been forwarded on the approach of Ontology development and how to describe services using these Ontologies.

As presented in Chapter 2, the widely discussed markup for Web Services, OWL-S, has presented an upper ontology describing the three essential aspects of a Web Service. Figure 3.1 [17] depicts the OWL-S Service Profile Classes and Properties

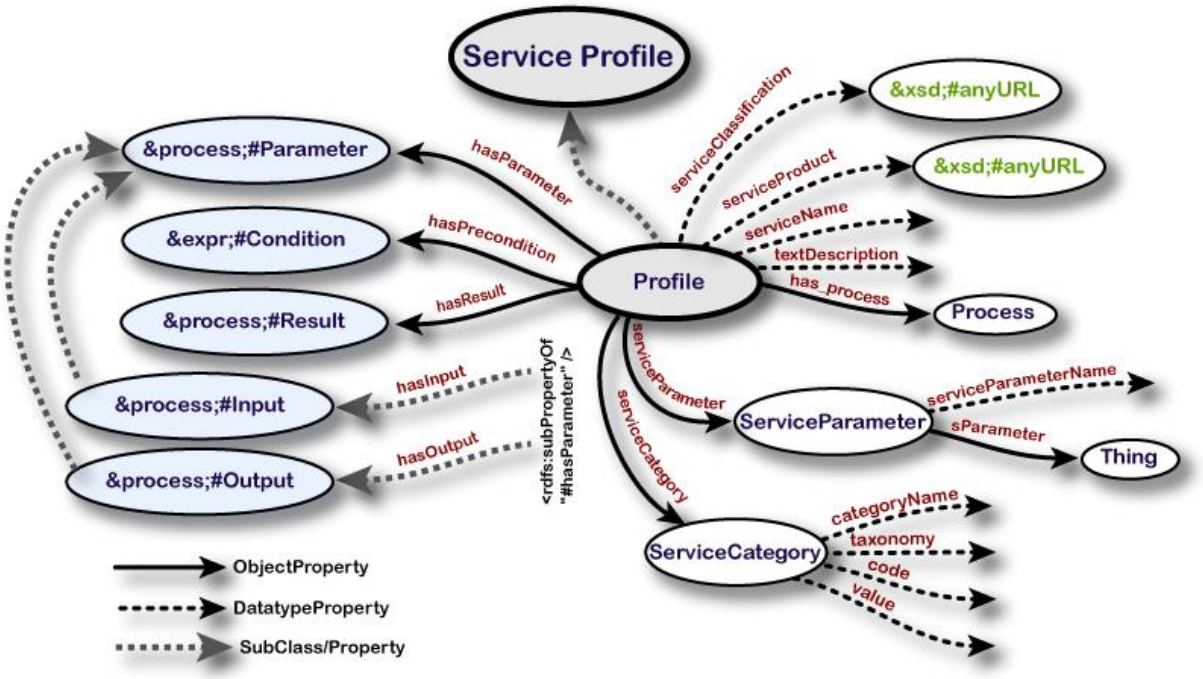

Figure 3.1: OWL-S Service Profile Classes and Properties

As shown in Figure 3.1[17], the Service Profile provides the utility to describe what a service does elaborately. The ServiceProfile described by a Profile provides the basic artifacts of a service including Name, the Publishers name and Address, Parameters (Input and Output), Preconditions and results.

As the OWL-S specification is complex and built in the top-down approach demanding for the complete change of the Web service framework, its applicability is limited while the W3C's recommended SAWSDL is getting momentum. After the adoption of SAWSDL by the W3C, the authors of OWL-S provide mechanisms to associate OWL-S profile with WSDL using the SAWSDL approach [51].

The W3C recommendation, SAWSDL, has just provided simple extensions for WSDL and XML schema to annotate WSDL components with any external Ontology. The work in [52] presented WSMO-lite service ontology to fill the SAWSDL annotations with concrete semantic service

descriptions. The WSMO-Lite addresses requirement of identifying the types and a simple vocabulary for semantic descriptions of services (a service ontology) as well as a language used to define these descriptions, define an annotation mechanism for WSDL using this service ontology, provide the bridge between WSDL, SAWSDL and existing domain specific ontologies. Like OWL-S, WSMO-Lite provides the utilities to describe the functional and non-functional attributes of a service in addition to the method of association with SAWSDL.

As noted in [53], the works so far focus on the formalization of generic web service concepts (like inputs and outputs). These works presented the approaches of semantic annotation of the different parts of service description using external ontologies in addition to the association of components among each other. In doing so, an external domain ontology is required that can be used for the annotations. There are works like [53] attempting to automatically construct domain ontologies through analysis of service descriptions. In all of the works, either the service ontology is assumed to be part of domain ontology or is left unaddressed. Usually domain Ontologies are general purpose and focus on real world concepts. In case of semantic web services, in addition to real world concepts there are artifacts defined by developers which may not exist in real world as a result may not be defined in domain ontologies. Furthermore, for robust and complete automated matchmaking and composition, as it is done for domain ontologies, instances services shall be captured as well.

3.4 Summary

The semantic Web service discovery, matching, composition and execution process remains a challenge. Recently, the use of semantics is being researched to assist conventional Web Services. But even with the help of ontology, machines are not yet able to discover and utilize services automatically. The developments in the area substantially use domain specific Ontologies assisted by Quality of Service computations – to find matching services and identify the most responsive service out of multiple candidates.

Primarily the existing approaches do not address the objective of automatic service matchmaking. Most of the works assume or expect the Consumer to request the service having all required data in advance and the negotiation between the Consumer and the Provider is limited. The primary focus of the majority of the works is a searching functionality for human users.

Significant number of works focuses on analyzing input and output parameters of advertised services for similarity against requested services. In reality, the similarity level of parameters will not guarantee similarity of capabilities. Services with similar input and output parameters may serve a completely different purpose. Other works consider matching based on the conceptual similarity of Service signatures (input, output and description) assisted by WordNet and ontology. This has also a major limitation as there is no universal naming convention. As discussed in [48], either parameter naming or descriptions are not regulated. Parameter and function names would be even hard for human understanding due to different naming styles, mix of words from different languages and abbreviations. There are a number of proposed Ontology languages to annotate services like OWL-S, WSMO and others. But, they are complex and demand fresh startup that needs a complete new set of standards and infrastructure and need developers to learn and abide by a new set of strict vocabularies (the ontology) – that obviously take away the flexibility of the developer and demands to work on a very complex set of ontology lexicons.

The major assumption in most of the works is the service consumer already searched and implemented during design time to an already known service. The proposed models attempt to find some other similar service to the implemented one in cases of failure. They have not considered the dynamicity of the Consumer's need. Cases like a Consumer looking for a service, at run time, that it has not interacted with before and do not have implementation details were not considered.

In our proposed model, we will work to address the automatic matchmaking requirement. The proposed model will enable consumers to describe their request using semantic annotations that will be used by the Broker to search from the available Services. When the Broker identifies potential service providers, the consumer will be informed of them. The Consumer will analyze the services and decide which one to use. In this process, the Consumer will negotiate with the Provider or interrogate itself to assess and determine inputs, trade-off constraints and pre-conditions. This model will enable service Publishers to provide additional information that will enable the Broker to better cluster services and draw service relationship graphs.

Chapter Four: Semantic Web Service Matchmaking Model

In Chapter 2, we discussed the Semantics Web service theories, tools and latest developments including the motivation driving this work. In Chapter 3, we presented the latest research works that attempted to investigate and address different issues that hinder from achieving the aspired goals. We also discussed the shortcomings of the Semantics Web Service, especially in discovery and matching, that this work will address. This section presents our proposed model.

4.1 Scenario of use

As demonstrated in [39], the Semantic Web and other technologies that emanated from it, strongly aspire for a strong human computer cooperation in which the current syntactical web progressed in to a huge inter-linked and meaningful data repository that machines can easily understand and use to perform important tasks on behalf of humans. In line with this vision, usage scenarios are presented that a users' software agent interacts and coordinates to achieve a goal.

Scenario I: Booking Transport Service (Multi-interaction)

The usage scenario of this model is where a user looks to book transportation from Addis Ababa to Gondar. This objective is submitted to the user's Consumer agent by a machine understandable, semantically annotated, description. The user's agent may not have served similar purpose before now and it may not know any transportation service provider to the specified location or even one that provides a transportation service at all. After receiving the request, the Consumer contacts the Broker submitting the same machine understandable query. The Broker has service providers clustered and linked based on their service similarity.

The Broker searches its provider's database and ranks these providers and their service offerings based on their functionality and reputation. While the Broker returns a couple of Providers, the Consumer analyzes candidate services based on the user's preferences like service quality and/or user's pre-set and historic preferences – means of transport, cost, date and so on. After the Consumer's agent chooses the preferred provider as per the user's preference, as the agent is authorized to use the user's personal information except payment details, which needs direct authorization of the user, the Consumer's agent dialogs with the Provider to finalize the service.

In the process of booking, the Provider requests for the User's personal details which the Consumer directly supplied while prompt the user for authorization of payment information. Finally, the Consumer books the transport to the preferred destination and informs the user. After successful booking, the Consumer records the service Provider's details for future use.

After a week the user prompts its agent to book a trip back to Addis Ababa. As the agent already knows a provider it attempts book his travel with its trusted Provider. Unfortunately, the Provider is not responding for unknown reason. Then the Consumer's agent contacts the Broker. The Broker knows some other service provider that provide the same service. Then the Broker informs the Consumer the new Provider, the Consumer then interacts with the new provider but this one requires the sub-city and street address of the user – which the previous Provider didn't requested and charges additional ten birr to transport the User from the Bus Station to the User's home. The agent prompts the user but the user drops the additional service. The agent then confirms booking without the additional service.

Scenario II: Currency Exchange (Single Interaction)

In another scenario, a web admin implements daily currency exchange information service from the National Bank of Ethiopia. It was working fine until this morning when it stopped replaying due to unknown reason. As the Web admin implements the service using a semantic web service, the Web admin's agent informs the Broker about the experienced challenge. Accordingly, the Broker replied the Commercial Bank's service which is almost similar with that of the National Bank and the admin's agent presents the information obtained from the new Provider. This case presents a service replacement process in which the actual human user who administers the Web site was not aware of the failure of the service as well as its replacement.

While the first scenario is a composite and needs several interactions to achieve the final goal, the second one is a straight forward and easy example. In both scenarios the main focus of this work is enabling enhanced service discovery and seamless service replacement that eliminates the need of direct human interaction in implementing the details.

The scenario of uses presented in I and II are practical examples that can be achieved with our proposed model. In current implementations, the Consumer need to engage to searching service

providers, identifying suitable service offerings and finally hardcode the implementation details. All these processes are done manually, with direct human intervention.

4.2 Design Requirements

The aspired objective of the Semantic Web and its derivative technologies are to enable machines understand the Web so that they can cooperate with human users. The Semantic Web is not a new form of Web but an extension of the current unstructured and dominantly syntactic Web, the same as the Semantic Web Service. The Web service has been around for a while and the technologies and tools have matured enough to draw significant interest in the industry. Like the Semantic Web aims to the Web, the Semantic Web Service aims at augmenting the Web Service by Semantics technology for better machine understandability and human machine cooperation. Derived from the general objective of the Semantic Web and Semantic Web Services [6, 9] and the Scenario of uses discussed in section 4.1, the basic design requirements of Semantic web services are: -

- **Machine understandability**: as discussed in previous chapters, the major requirement of the semantic based technologies is to enable machines understand and act autonomously to perform some task and cooperate with human users. Derived from this requirement, the major design requirement for our model is machine understandability. When we say machine understandability, it is to mean that, machines should be able to analyze the description of services and requirements of users. It is to provide as much information as required and as per agreed ontology so that the machine is able to process the information.
- **Run-time discovery and matching**: unlike the current implementation of web services which the developer is expected to manually discover and hard code the implementation of services, the other major requirement of this model is to determine and discover services at run time and consume it providing required information. This implies that, the Consumer's developer may not know the existence of the services, its location and implementation details during development. It is expected to dynamically discover services that can satisfy the requirements of the user and bind on run time through negotiation. It requires the Consumer's agent to understand the queries of the user and find the services that can achieve it.
- **Service replacement**: in line with run-time discovery, service replacement is another key requirement in which a Consumer that was using one of the available services should be

able to use any of other available similar services in case of failure or preference. With a matured and trusted implementation of the above run-time discovery this requirement would be less important, as all discoveries and matching happen at run time, but in major cases consumers develop targeting some specific implementations and Providers. As a result, this requirement is important for such targeted implementations to dynamically discover and match to other similar services – when the implemented one has failed or for some other reason.

- **Dynamic binding through negotiation**: Services need inputs and provide outputs and have constraints. Different implementations of a service may have different requirements and constraints while serving same or near same purposes. The current implementations require the developer to know the exact implementation details of services and code abiding by the details. This has a great barrier to the above requirements of run-time discovery, matching and replacement as the developer won't possibly know and cannot code all possible alternative implementations. Derived from this, the major requirement of this model is dynamic binding through negotiation, in which the Consumer will deal with candidate Providers on the terms of use – inputs, outputs, constraints and other factors. The Consumer shall be leveraged from knowing these implementation details ahead of use.
- **Multiple Semi-autonomous agents:** the above basic requirements envisaged that the Consumer and Provider are capable of dealing autonomously and learn through time at some extent. The Consumer is expected to decide and act by its own to decide which service to use among candidate services through negotiation and dialogue with the Provider- after receiving a goal from the user. The dialogue may not be with just one Provider but with several until it finds a suitable one. Therefore, the environment is depicted as an interaction of multiple autonomous and semi-autonomous agents.

4.3 Assumptions

In designing this model, the following key assumptions are made.

- There is a shared ontology (both domain and service ontology) among the three actors (the Provider, the Broker and the Consumer) or there is a mapping Ontology otherwise.
- The developer provides enough description of the service as per the SAWSDL standard and using a shared domain and service Ontologies. Inputs and outputs are annotated using

either domain or service Ontology and Service description, preconditions and results are annotated using the service Ontology.

- The Service relationship and dependency graph (Service-Net) is not a one time and one-man effort. It is assumed that a publicly available service network (service grid) will be available. This work outlines the backbone and startup only.
- The model uses the existing Web Service messaging protocols and SAWSDL standard.

4.4 Architecture

Discovery and matching is a critical process for Web service utilization. A service that cannot be discovered and matched cannot serve its intended purpose. Since the inception of Web services, discovery and matching processes have been labor intensive and require direct developer's engagement. To our knowledge, the development in the area has not yet fully or significantly automated the process.

The proposed model maintains the basic architecture of Web service while enhancing each stakeholder with additional components that provide intelligence. Our architecture is depicted in Figure 4.1.

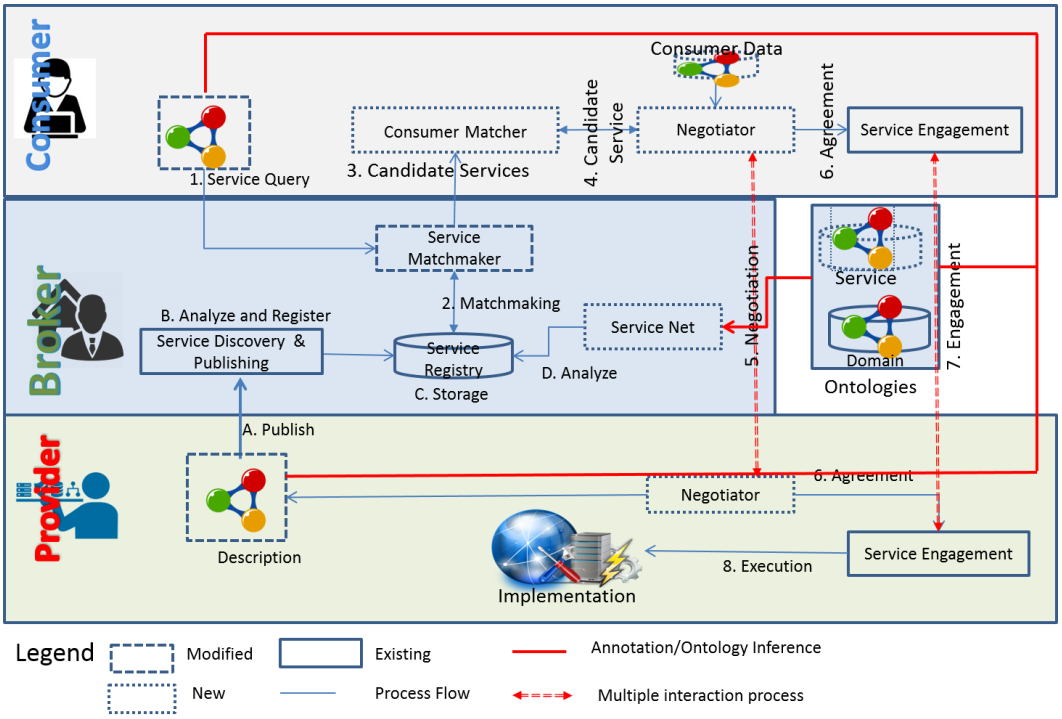

Figure 4. 1: Proposed Architecture

- **Publishing (flow)**

A. The Provider implements its service and prepares annotated service description using the domain Ontology (both service and data) and publishes to the Broker.
B. The Broker validates and analyzes the description.
C. The Broker stores the service description in the service registry.
D. Finally, the service description is analyzed and ranked in the service network for discovery (at the broker).

- **Discovery and Matching flow**

1. The user constructs its query and submits to the Broker. The query interface at the Broker validates the query according to its semantic rule and against the shared Ontology.
2. The Matchmaker traverses the service registry assisted by the service-net information for candidate services.
3. The Broker's matchmaker provides to the Consumer's service matcher component appealing candidate services.
4. The Consumer's service matcher component analyzes the candidate services based on its internal policy for service selection. It identifies the candidate service and notifies the negotiator for further negotiation.
5. The negotiator corresponds with the Provider's negotiator for agreement. This may have several back and forth correspondence (shown in arrow at both ends),
6. When the negotiators arrive on agreement, they will notify their respective service engagement components about the deal and the terms. If the two negotiators fail to come to consensus, the negotiator notifies the Service matching component to get the next matched service Provider. The flow resumes from number 4.
7. The service engagement component interacts to deliver the agreed service. The Consumer provides the required data as per the earlier agreement.
8. The Provider contacts its internal implementation for execution and returns the result.

4.4.1 The Service Provider

The service Provider implements functionalities and exposes its interfaces in a form of Web services to the rest of the world. The Provider is the actual implementer of services. Regardless of

the implementation language and platform, the Provider exposes its implemented services using standardized and vendor neutral descriptions as per the specification of SAWSDL. The underlying communication and consuming procedure is provided by the Web Service protocol.

The traditional Web Services provides service descriptions in a standardized WSDL format. The WSDL provides a means of expressing a Service's inputs, outputs and addresses in a uniform and platform independent format. Service Brokers use this information to build a service registry for searching and binding. The Semantic Web Service adds semantics to the WSDL through semantic annotation, while maintaining the foundation XML and SOAP protocols. In Figure 4.1 the consumer lane shows our proposed architecture of the Provider.

4.4.2 The Service Broker

The Service Broker is the middleman that facilitates and enables the communication between the other two stakeholders. It is a central registry of services where service providers publish services and Service consumers query to find machining services. It is commonly called as UDDI. From its inception, it has been XML based, standard and open. While the traditional UDDIs were searched using keywords their semantic counterparts have semantic information. This component is the brain of the model. It includes features to publish and discover services, analyze requests and services, query engine and a service repository database. As outlined in the requirements, our model splits the process of matchmaking between the traditional owner, Broker, and the Consumer. Selecting one service out of potential candidates is shifted from Broker agent to the Consumer agent.

The Broker agent encompasses the Discovery and Publishing, Service Registry, Service Net and the Matching engine component. The Broker lane in Figure 4.1 depicts the architecture of the Broker.

4.4.3 The Consumer

The Consumer is the seeker of functionality or service from the Web. As per the current implementation, manual service selection and engagement, the Consumer's developer needs to identify the service that fulfills the requirements and manually code the implementation. As a result, Consumers are usually embedded deep into the consumer's other, none service, business logics.

As outlined in the model's requirement, we recommend a general purpose self-contained and intelligent Consumer agent that can perform tasks autonomously. With its autonomy and intelligence, given a query or a goal by a human user, it should be able to deliver or perform the requested task by corresponding with the other stakeholders. To support these goals our model has Query Interface, Service Matcher, Negotiator, Consumer Data and Service Engagement components. The Consumer lane in Figure 4.1 depicts the architecture of the Consumer agent.

4.5 Ontologies

In addition to the four common stakeholders, our model incorporates a resource for semantic annotation. An Ontology shared among the three actors is recommended to annotate Service Descriptions, User Queries and Consumer data. In cases where a shared ontology does not exist there must be translation Ontology to interoperate Ontologies from different sources. Our model recommends a new form of domain ontology called Service Ontology defining programmatically defined services and associated IOPRs in addition to the domain ontology.

One of the major challenges of the Internet is its unmanageability or there are no rule governing authors on the Internet. Developers or Providers have every right to follow their own convention in authoring content as well as naming services and IOPRs. The SAWSDL chooses in favor of this freedom over a naming convention. As a result, SAWSDL is a bottom-up approach of web service annotation using concepts from domain ontologies. As the main component of the SAWSDL specification and our model, Services and their IOPRs will be semantically annotated from concepts of their respective domain ontologies.

In an extension to SAWSDL specification which assumes the use of domain Ontology for annotating services, we propose a separate and specialized Web Service Ontology (WSO) on top of the domain data semantics (domain ontology).

4.5.1 Service Ontology Language

The reason for defining a specialized Service Ontology is because the scope of domain Ontology is solely on the defined domain and it is multipurpose. Mixing up service taxonomy with domain taxonomies will make the Ontology huge, complex and difficult to maintain and use. In addition, the service taxonomy is used rarely as the ontology is general purpose and most of the time, its

uses are for domain concepts. Therefore, we proposed a separate specialized ontology called Web Service Ontology that formalizes service names, pre-conditions and results of Web Services.

As domain Ontologies are a reflection of real world concepts, they may lack the flexibility and complexity that are needed by developers. Often inputs and outputs are defined in their respective domain Ontologies as they are a reflection of real world concepts. But complex or specialized input and output that are defined programmatically will not be defined in domain Ontologies a they are not known to domain concept experts. As a result, we provide the flexibility to define Inputs and Output that are not common in the domain to be defined in the WSO. Like any other domain Ontology, Web Service Ontology is developed and maintained either manually or semi-automatically. It is a specialized domain Ontology that specifies the Services in a given domain, their hierarchical and functional dependence and associated IOPRs. The Web service ontology is a specialized domain ontology that formally defines the services, inputs, outputs, preconditions results and relationship of the domain services.

In following this approach, the domain Ontology remains focused on real world concepts of the domain and the attached specialized Web service ontology is used to define the services of the domain. The main argument behind Web Service Ontology is, we cannot possibly impose a naming convention on service developers and authors. The Web Service Ontology provides publishers the taxonomies to annotate their services. The annotation provides the very much needed information in terms of similarity and dependency at the same time respecting the freedom of developers for naming their services.

The main concepts of the service ontology are service naming and description, functionality, conditions and Results. Each concept is presented here in below.

1. Service Name and Description

As noted above SAWSDL has not imposed any naming convention or guideline. As our service ontology is proposed to draw a common understanding among developers in a specific domain, the service name and description sub-ontology will enable a shared library of services names and human understandable description for a given domain. The service names under this ontology are purely for annotation purpose after the implementers gave their own preferred names for their own instances.

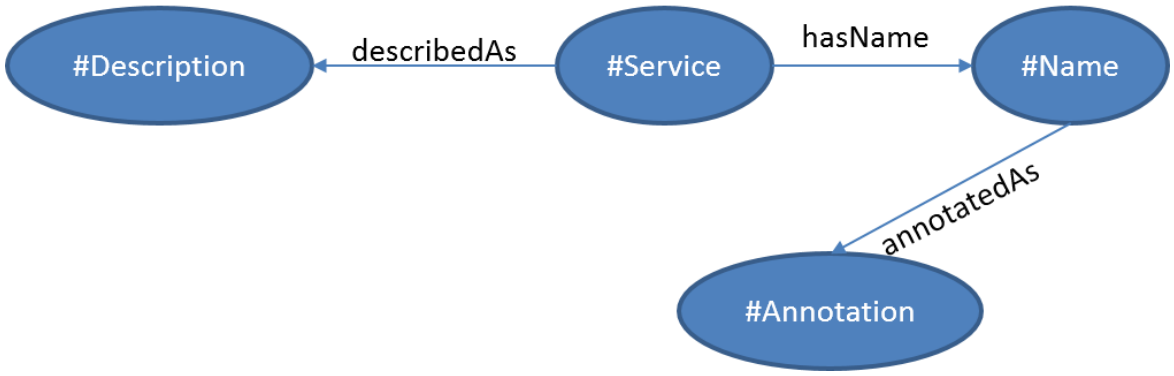

Figure 4.2: Service Name and Description sub Ontology

As shown in Figure 4.2 service has ServiceName and Description with properties hasName and describedAs linking them to the service respectively.

- ServiceName: - a common and meaningful Service name as defined by ontology developers
- Description: -Textual description of the service – what it can do. The primary audiences of this are human readers and NLP based clustering agents.
- Annotation is the reference service name from the Web Service Ontology (WSO)

2. Functionality

A service exists to provide some functionality. Functionality is the promised purpose that the service offers. Describing this functionality is important for both manual and automated matchmaking. Figure 4.3 presents our proposed functionality sub-Ontology.

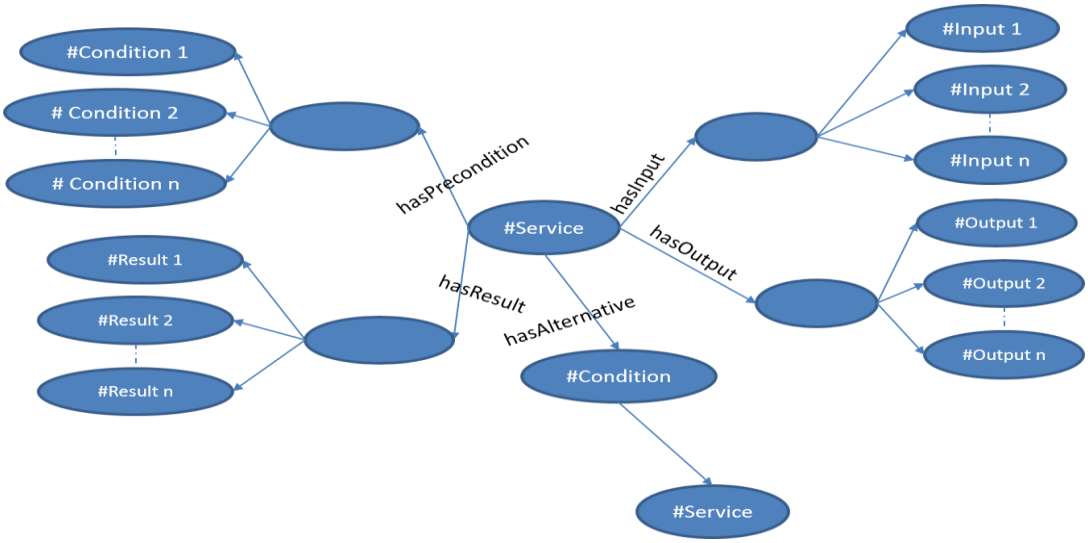

Figure 4.3: Functionality Description sub Ontology

- **Input (hasInput)**: - inputs are data/information expected from the Consumer. Inputs need to be described properly and annotated semantically to enable the Consumer understand what really is/are the required data.
- **Output (hasOutput): -** like inputs, services have one or more outputs. Some services perform some task/computation and provide the output, information, to the consumer. Others perform some task and notify the consumer of their action and while others do both. In case of the first type of services, output is critically important- the main purpose of the services is to provide the outputs. The example of this is tax calculator which takes the gross salary and returns the tax amount and the net salary. In case of this, the capability of the service is expressed in terms of the output. While the other two are not. Though describing the output at same depth like the input is important for all scenarios, it has more significance in case of services of type one.

- **Precondition (hasPrecondition)**: - this attribute defines the conditions that must be met to use the service. A service may or may not have a precondition and can have more than one precondition.
- **Result (hasResult): -**also called effect is the impact of the service on the real world. Services will have impacts to the real world like a hotel room is booked or air-ticket is issued. In cases of these operations the output is merely the notification of the result and the real world effect is not expressed. This result indicates the missing attribute of the output to the real world, impacts.
- **Condition (hasAlternative)**: - while the IOPRs provide the success scenario or the primary use case, the conditional attribute provides alternative/conditional flows that can be determined based on the needs of the Consumer. The conditional attribute has sets of parameters and results based on the conditions. A service may or may not have one or more conditional flows.

3. Service Relatedness Sub Ontology (relatedWith)

Generally, services are not isolated or independent or are not unique. There are many more similar, exactly the same or partially similar services. During service publication, Publishers or service ontology designers specify the relatedness of a service with other already existing/known services.

This service relatedness is expressed in terms of similarity with other services. Figure 4.4 shows the service relatedness sub-Ontology.

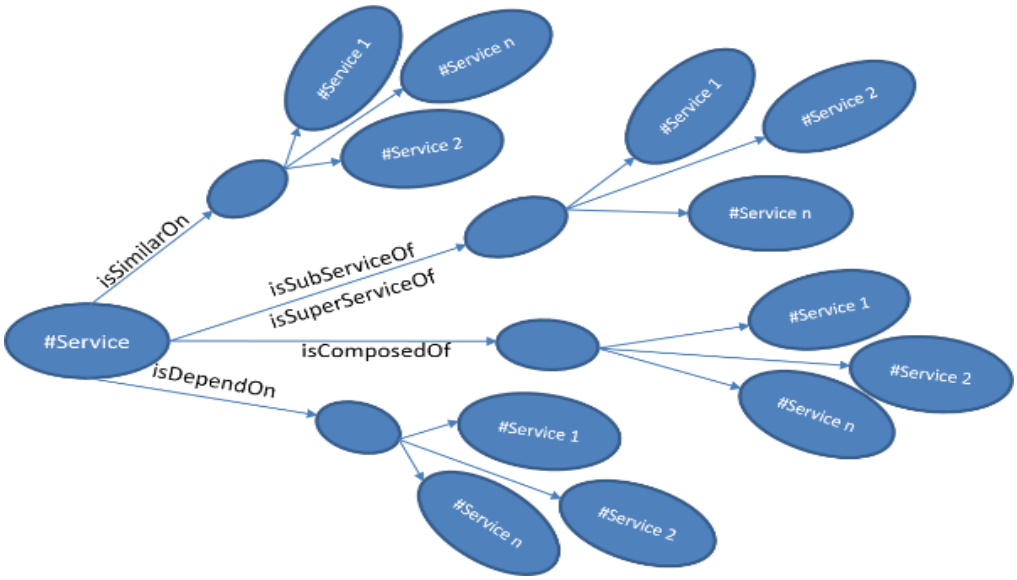

Figure 4.4: Service Relatedness sub Ontology

- **Subservice (*isSubsServiceOf*): –** a service could be a specialization of another wider scope service.
- **Super Service Of (*isSuperServiceOf*): –** inverse of a subservice of relation that a service provides more general and bundled functionalities that are provided by its sub services. The parent class of all services is "Service".

The case demonstrating the above two relations is the travel booking scenario discussed in the scenario of use section. In the travel booking example, a single travel booking service may do all the necessary bookings for the travel from Addis Ababa to Gondar including the taxi service from home to Bus Station, the actual travel from Addis Ababa to Gondar and a hotel reservation in one of the Hotels in Gondar. While there could be specialized services for tax hauling, bus and hotel booking separately. The first service is a super service to the other specialized three and the three are sub services of the first one.

- **Composed Of (*isComposedOf*): –** in a similar way a service could provide services provided by a set of other services. Composed Of is a special type of sub service relation in which the super service uses the sub services somehow to deliver its

functionality. This information is to specify composite services with the composition logic and associated services. In the case of Composed Of the service is provided by the execution of these individual services by some condition, sequence or choosing.

In the same example mentioned above, there could be another service that orchestrates the other three specialized services (taxi hauling, bus and hotel booking) to act like the super service. The Composed of property enables to capture such cases.

- **Similar With (*isSimilarWith*):** – this attribute associates the service with other services that provide similar functionality. It applies to all classes Services and IOPRs but the domain and Range shall be same class. It is both symmetric and transitive.
- **Depends On (*isDependsOn*)**– services may depend on another service/s to deliver their functionality. This attribute provides list of services that the service depends on. The difference from Composed Of is in the former case the service partially depends on the other services while it is an orchestration in the later case.

4. Condition sub-ontology

Similarly, conditions, that we use to specify preconditions or conditional executions, have relations with each other. Figure 4.5 shows the properties of conditions.

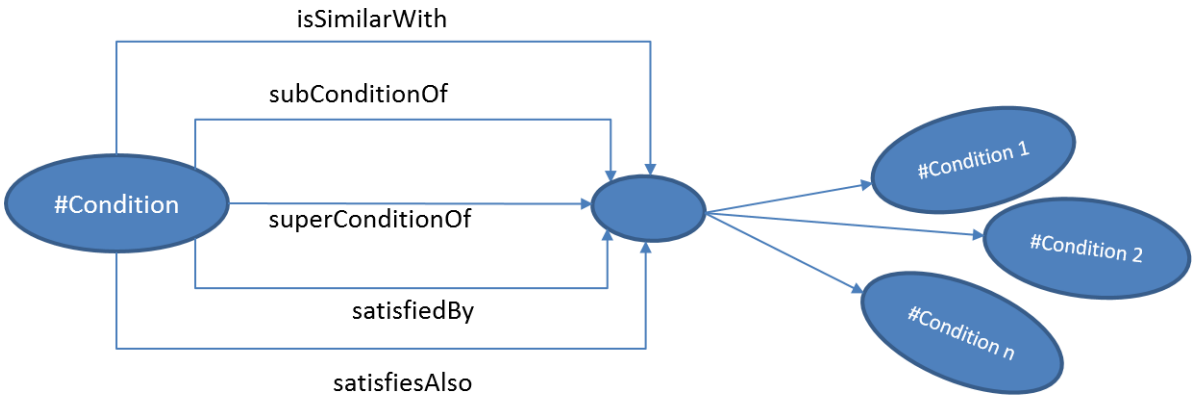

Figure 4.5: Condition sub Ontology

- **Satisfied By (*satisfiedBy*): -** This property specifies the dependence/relatedness of conditions. Satisfying one precondition may mean either satisfying another one or checking the other one is unnecessary. For example, if we confirm that the user has

sufficient funds in his/her account, it may not be necessary to check the validity of the credit card. This is a directed relationship from the later one to the already validated and satisfied precondition. The inverse may not be necessarily valid.

- **Satisfies Also (*satisfiesAlso*)**: - The inverse relation of satisfied by. It is an implication that if the currently validated precondition is satisfied the other is also satisfied.
- **Subclass/ SubConditionOf**: - a hierarchal relation where one is a general concept of another. It applies only for conditions.
- **Super Condition of**: - the inverse of Sub condition of.
- **Similar with (*isSimilarWith*)**: - when two conditions are identical

5. Result/Impact Sub-Ontology

Results/Impacts are the bi-product of outputs that has direct impact to the real world. Similar to conditions and others, results have also similar relationship with each other. Figure 4.6 presents the result sub Ontology.

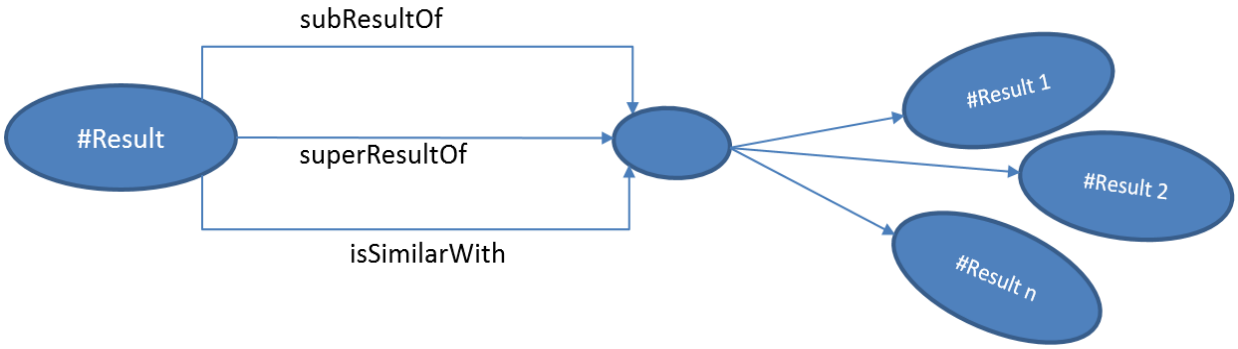

Figure 4.6: Result sub Ontology

- **Subclass/ Sub Result (*subResultOf*): -** a hierarchal relation where one is a general concept of another. It applies only for results.
- **SuperClass/Super Result (*superResultOf*): -** the inverse of Sub Result Of.
- **Similar With (isSimilarWith): -** one Result is similar with another.

The following depicts the service ontology representing the above concepts.

- **Classes**: - We have 4 main classes namely Service, Condition, Result and Parameter. Input and Output are subclasses of the Parameter class. These classes are defined as follows.

```
<owl:Class rdf:about="Service"/>
```

```
<owl:Class rdf:about="#Parameter"/>

<owl:Class rdf:about="#Condition"/>

<owl:Class rdf:about="#Result"/>

<owl:Class rdf:about="#Input">
   <rdfs:subClassOf rdf:resource="#Parameter"/>
</owl:Class>

<owl:Class rdf:about="#Output">
   <rdfs:subClassOf rdf:resource="#Parameter"/>
</owl:Class>
```

- **Properties**: -Most of the time inputs and outputs could be expressed in the general domain ontology. The following shows the definition of properties (input, output, precondition, result and others).

```
<owl:ObjectProperty rdf:about="#hasInput">
   <rdfs:subPropertyOf rdf:resource="#hasParameter"/>
   <rdfs:domain rdf:resource="Service"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="#hasOutput">
   <rdfs:subPropertyOf rdf:resource="#hasParameter"/>
   <rdfs:domain rdf:resource="Service"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="#hasPrecondition">
   <rdfs:domain rdf:resource="Service"/>
   <rdfs:range rdf:resource="#Condition"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="#hasResult">
   <rdfs:domain rdf:resource="Service"/>
   <rdfs:range rdf:resource="#Result"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="#dependsOn">
            <rdfs:domain rdf:resource="Service"/>
            <rdfs:range rdf:resource="Service"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#subServiceOf">
   <owl:inverseOf rdf:resource="#superServiceOf"/>
   <rdfs:domain rdf:resource="Service"/>
   <rdfs:range rdf:resource="Service"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="#superServiceOf">
    <owl:inverseOf rdf:resource="#subServiceOf"/>
   <rdfs:domain rdf:resource="Service"/>
   <rdfs:range rdf:resource="Service"/>
```

```
</owl:ObjectProperty>

 <owl:ObjectProperty rdf:about="#satisfiedBy">
    <owl:inverseOf rdf:resource="#satisfiesAlso"/>
    <rdfs:domain rdf:resource="#Condition "/>
    <rdfs:range rdf:resource="#Condition "/>
  </owl:ObjectProperty>

    <owl:ObjectProperty rdf:about="#satisfiesAlso">
    <owl:inverseOf rdf:resource="#satisfiedBy"/>
    <rdfs:domain rdf:resource="#Condition "/>
    <rdfs:range rdf:resource="#Condition "/>
  </owl:ObjectProperty>

  <owl:ObjectProperty rdf:about="#subConditionOf">
    <rdfs:domain rdf:resource="#Condition "/>
    <rdfs:range rdf:resource="#Condition "/>
  </owl:ObjectProperty>

  <owl:ObjectProperty rdf:about="#subResultOf" >
    <rdfs:domain rdf:resource="#Result"/>
    <rdfs:range rdf:resource="#Result"/>
  </owl:ObjectProperty>

 <owl:ObjectProperty rdf:about="#similarTo">
    <rdf:type rdf:resource=" #SymmetricProperty"/>
    <rdf:type rdf:resource="#TransitiveProperty"/>
</owl:ObjectProperty>

 <owl:ObjectProperty rdf:about="#sameAs">
    <rdf:type rdf:resource=" #SymmetricProperty"/>
    <rdf:type rdf:resource="#TransitiveProperty"/>
</owl:ObjectProperty>
```

6. **Sample Web Service Ontology**

Sample 1: Trip Scheduling Ontology

Illustration of the super and sub service of properties is a travel scheduling service. One very general service may have bundled all trip scheduling services like air ticket booking, taxi service, hotel, tour guide and other services in one service while other specialized services may exist like one service could provide only air ticket booking another hotel reservation and another to contract a tour guide. In this scenario we state that the first general service is a super service of the other specialized services and the specialized services are sub-services of the general service. Figure 4.7 presents the excerpt of the service ontology for the above scenario.

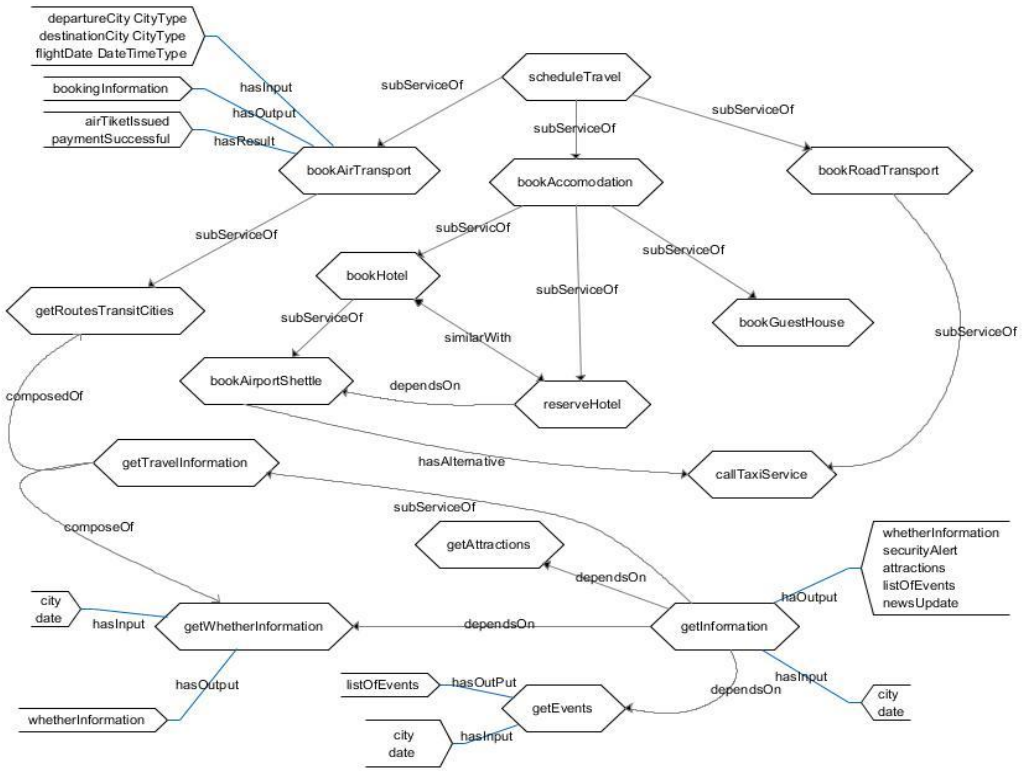

Figure 4.7: Sample Travel Schedule Web Service Ontology

```
<owl:Class rdf:about= "#scheduleTravel ">
  <rdfs:subClassOf rdf:resource ="#service"/>
</owl:Class>
<owl:Class rdf:about= "#hotelBooking ">
  <rdfs:subClassOf rdf:resource ="#tripScheduling"/>
</owl:Class>
<owl:Class rdf:about= "#contractTourGuide">
  <rdfs:subClassOf rdf:resource ="#tripScheduling"/>
</owl:Class>
```

7. Usage of Service Ontology with SAWSDL

As expressed in the above section, Service ontology is a formalization of services in a given domain. Its development is mainly manual with limited or no support of automation. The main purpose of its existence is to enable developers annotate their services using this shared ontology. The SAWSDL provides the utilities to enable users to annotate their services using any given ontology. The following shows how to use our proposed Service Ontology in SAWSDL service description.

```
<wsdl:message name="get_DESTINATIONINFORMATIONRequest">
   <wsdl:part name="_DESTINATION" type="CityType"/>
  <wsdl:part name="_DEPARTURE" type="CityType"/>
  <wsdl:part name="_DEPARTUREDATETIME" type="DateTimeType"/>
</wsdl:message>
```

```
<wsdl:message name="get_DESTINATIONINFORMATIONResponse">
   <wsdl:part name="_DESTINATION" type="CityType"/>
  <wsdl:part name="_ARIVALDATETIME" type="DateTimeType"/>
  <wsdl:part name="_DESTINATIONWHETHER" type="WhetherType"/>
  <wsdl:part name="_TRANSITCITY" type="CityType"/>
</wsdl:message>
<wsdl:portType name="DestinationSoap">
   <wsdl:operation name="get_DESTINATIONINFORMATION">
<wsdl:input message="get_DESTINATIONINFORMATIONRequest"/>
<wsdl:output message="get_DESTINATIONINFORMATIONResponse"/>
   </wsdl:operation>
</wsdl:portType>
```

The above get_DESTINATIONINFORMATION service presents a business case for a travel agency in which the Consumer provides the destination and departure cities and the departure date and time to get the destination city's whether condition, estimated arrival date and time, and the transit cities. The service is annotated with appropriate ontologies with the type information. CityType, DateTimeType and WhetherType are concepts defined in the service and annotated with the sawsdl:modelReference. The above example is a service which exactly looks WSDL but SAWSDL service description. Note that the CityType, DateTimeType and WhetherType are defined and annotated in same file.

In the above example, a first time user or a user without accompanying description will not know what purpose get_DESTINATIONINFORMATION could serve. The clue provided by its name is very limited and ambiguous. The agent which receives this service will also have difficulty to analyze and categorize it with the inputs and outputs information.

In our model, in addition to the inputs and outputs, which are already annotated in the above case, the operation will be annotated against domain Web service ontology. In our example, the service description will be annotated with the ontology provided in Figure 4.7: Sample Travel Schedule Web Service Ontology

```
<wsdl:operation name="get_DESTINATIONINFORMATION"
sawsdl:modelReference="http://.../TravelOntology#getTravelInformation">
  <wsdl:input message="get_DESTINATIONINFORMATIONRequest"/>
  <wsdl:output message="get_DESTINATIONINFORMATIONResponse"/>
</wsdl:operation>
```

As it can be easily noticed, the get_DESTINATIONINFORMATION is not part of the ontology while getTravelInformation serves the required purpose. In other terms the former is the service name provided by the developer while the latter is a preferred name by the ontology developers. The

ontology provides rich information that is useful for matchmaking apart from the service name annotation. As it is clearly presented, getTravelInformation is a sub service of getInformation and it is a composition of getWhetherInformation and getRouteTransitCities. In case getTravelInformation is not available, the Consumer can use getInformation with a little inconvenience of additional information (security alert, attraction sites, events and news update) or it can invoke getRouteTransitCities and getWhetherInformation to get exact replica of getTravelInformation.

4.5.2 Service Profile Ontology

The service profile/description provides the required information by both Consumer and Broker agents for matchmaking and negotiation. Describing services exhaustively is one of the key challenges that hinder the matching process. As assessed in Chapter 2, automatic matching requires a clear service description beyond the traditional textual and keyword based descriptions.

For our proposed model, service description is a key enabler of the model's intelligence and dynamicity. As clearly described in Chapter 2, the SAWSDL standard does not provide any language or top-level ontology for semantic service description. It just provides the mechanisms of annotating existing WSDL components from ontology defined somewhere else. On the other dimension the WSDL is well known for its mere syntactic nature.

WSDL and consequently SAWSDL lacks attributes that would enhance the capabilities of matchmakers. For example, if a Service provider has a service which takes a list of numbers and returns their sum, as per earlier works, the Provider can only publish the service name, inputs, outputs with human understandable description while the Semantic counterparts do the same with semantic information. In our model, Providers can publish additional information that can assist either in service discovery or negotiation process. This includes the services similarity with other already published services, negotiable specifications – if the Provider has flexibility how to use the Service and other specifications that help the Consumer to decide. Our proposed service description is an extension to the OWL-S's service profile through the addition of specialized classification attributes and conditional process flow. Figure 4.8 shows the service profile description classes and properties.

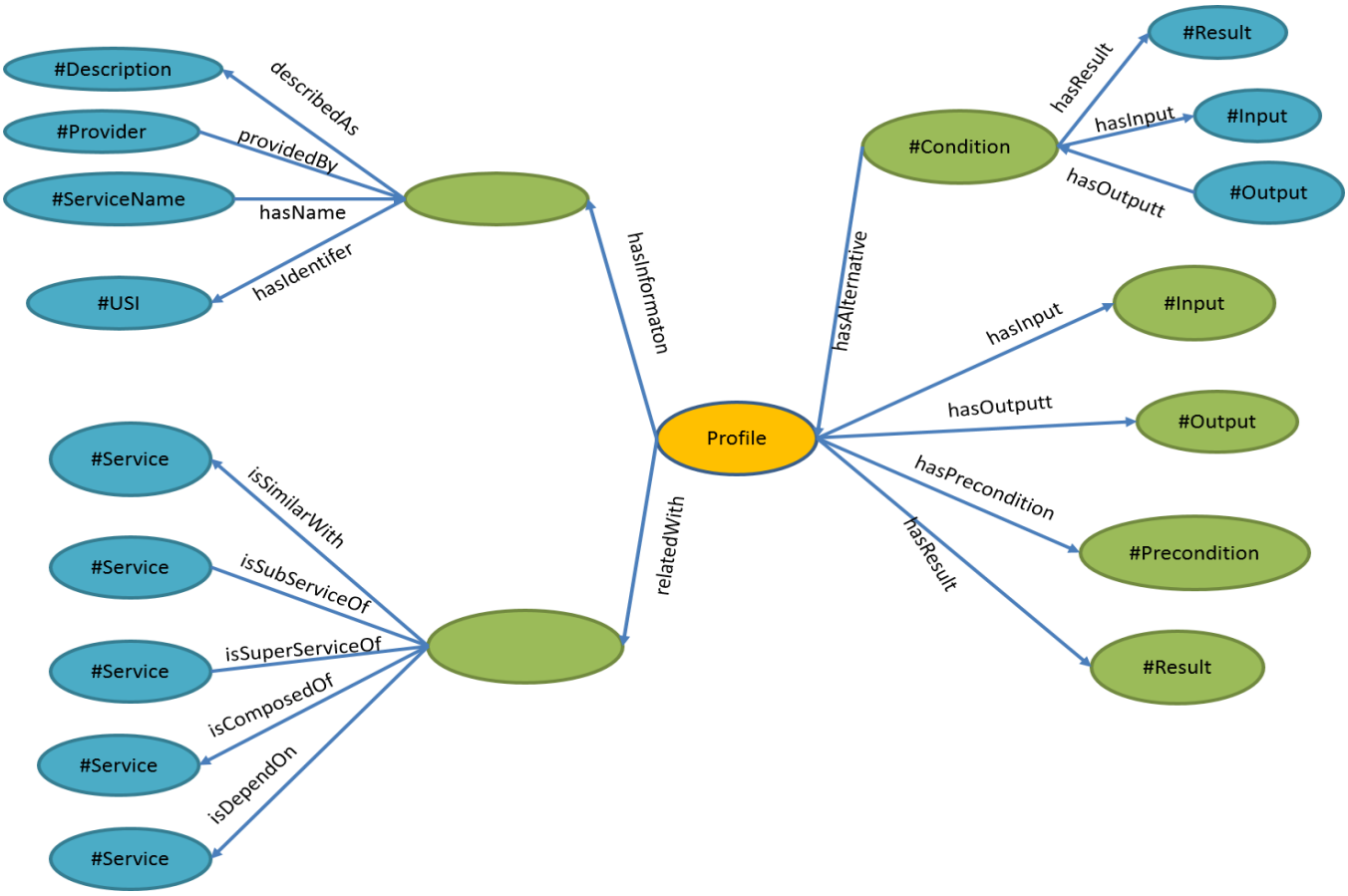

Figure 4.8: Service Profile Classes and Properties

Note: the purpose and usage of the classes and properties of service profile are exactly as discussed in Section 4.5.1 (Service Ontology Language).

The current SAWSDL based infrastructure does not support submission of service profile/description as per our proposed model. Accordingly, the workaround is to submit a separate service profile constructed as per our approach to the Agent. Alternatively, this service description can be added to WSDL as the specification does not actually block adding additional information or content on the standard WSDL as long as it is XML compliant.

The Service Ontology Language and the Service Profile Ontology are used by Service ontology developers and other interested developers who want to express their service exhaustively. In normal circumstances instances of services are just needed to be annotated using the SAWSDL model reference approach against the provided ontology. The Service-Net (Section 4.6) will be constructed based on similarity of instance services.

4.5.3 Service Discovery and Publishing

This component has a bi-directional function. It enables service Providers to publish services as per agreed standard description format and it also navigates potential service providers to collect service descriptions. In both cases, this component provides rich set of attributes that enable providers to describe the service capabilities at better depth. While the primary approach is the publish approach, the discover approach will also work for known Providers and service registry exchange with other Brokers.

The key starting point of the process is a well-structured and annotated service description. The service description is the one that makes the major difference.

After the service description is constructed, the service Provider shall submit the description to the Broker. This is called publishing. The Broker parses the description to check its compliance to the service specifications and the ontology. The Broker may also traverse known addresses regularly to check if new services are advertised or check if known services are still functioning through the crawling function.

4.6 Service Similarity Network (Service-Net)

Service Ontology is a specialized domain ontology which defines the possible services in a given domain. The ontology's definition will depend on the understanding of the ontology developers. Service Providers, who are relatively large in number and different from the ontology developers, actually develop their services and annotate using the provided ontology. The ontology developers provide the abstract definition of services while the Providers actually provide the concrete services, instances of the Ontology concepts.

Although the ontology provides an abstract definition of a service, name, inputs, outputs, preconditions and results, Providers have the freedom of implementing a service as per their desired approach and requirement. As such, it is not straight forward to determine the similarity of instance services even if they are annotated against the same concept/class on the ontology.

Measuring the level of similarity of instance services goes beyond the annotations of service profiles against some ontology. A service-net (Service Similarity graph) based on similarity information, that links service instances based on their similarity or relationship information will be constructed by this component. The final output of the service-net sub component is a linked

and interconnected service network based on the functional similarity of services. Service similarity graph is a weighted graph from Service S to S' with a weight value computed from the functional similarity of the services connected by the edge.

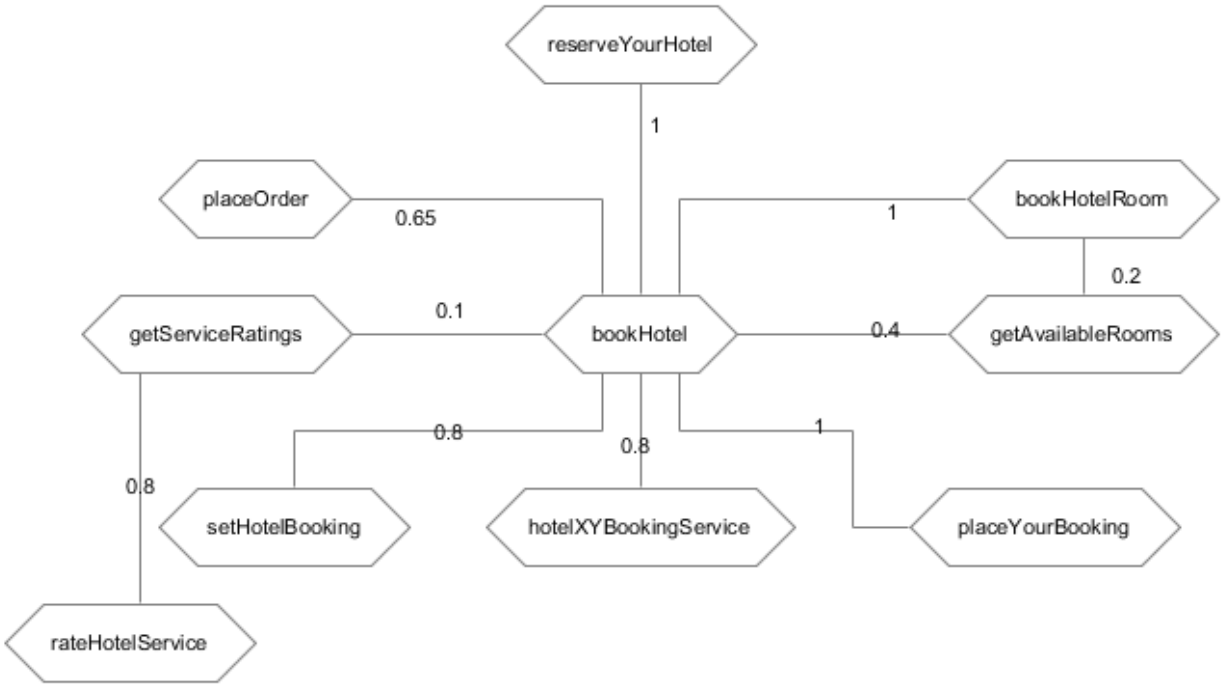

Figure 4.9: Sample Service Similarity Network (Service-Net)

Figure 4.9 presents a sample Service Net as the weight (the similarity level of two services) is computed as per Algorithms and Equations that will be discussed in Section 4.6.1. Assume that the services shown are published by different providers to a common agent annotated using the same ontology concept *bookHotel*. The names in the hexagons are the given names by the providers while all are annotated against the same concept *bookHotel*. Note that all providers except the one placed at the center use their own naming style.

Despite the common concept they are annotated against, the actual functionalities of the services are not clearly known. We compute the similarity of instance services and construct a graph connecting soundly similar services for two major reasons: -

1. Service replaceability: - as we discussed, service replaceability is an important phenomenon in service matching. Service Agents shall be able to respond to consumers with alternative services when the one known to the Consumers is not functioning or when needed.

2. Efficiency of matchmaking process: - other models compute the similarity of published services against the service query. This approach is inefficient as it needs to compute the similarity of the query against all existing/published services without prior knowledge and direction of searching. Computing the similarity level of instances will greatly assist the matchmaking process.

4.6.1 Similarity Computation

Service similarity is the measure of the functional similarity of two Services. The level of similarity ranges from exactly the same to completely disjoint. Exactly same services function the same way, partially matched services provide the partial solution of a service while others are completely disjoint. Service similarity measures the functionalities similarity level of instance service while the implementation details and solution approach may vary from Provider to Provider.

Measuring service similarity is critically important to determine replaceability. Determining similarity and replaceability would be relatively easy for informational services while it would be challenging for functional services. The first kinds of services, informational or computational, perform some computation or extract some information and respond to the Consumer. They are purely informational and there is no significant post-condition (result) that will happen as the result of the process. While for the second kinds, functional, they perform some task and inform the post condition (result) to the consumer. The post condition would be persistent or may have further consequences.

As the first kinds of services are purely informational, usually there is no persisted information like authentication or membership, which affects the user's instantaneous access. News services are the practical examples of this kind. The user's need in accessing a news service is obtaining updated information –either today's headlines, breaking news or the user's preferred issue. The user may have a preferred news source but in case the preferred source does not work, the user won't mind receiving updates from another news service provider. As a result, despite user preference issues to consider, informational services are replaceable to one another.

On the other hand, functional services are challenging to generalize in terms of replaceability. For instance, we may categorize social network services under the same category as they provide the

same service of status update, profile view, sharing photo, send birth day notification and so on. While the services are meant for almost same functionality, it is not like the informational services as the user needs to have previous engagement and information with the Providers. In practical terms, a user cannot post a picture to his/her Facebook wall unless the Facebook photo posting services is functioning, it cannot be replaced with similar service from Twitter. On the other hand, many services do not need similar information or engagement history to use their services. The practical examples are map or e-commerce services. Though the user may prefer one Provider over the others, almost all Providers accept similar information (longitude and latitude) and provide exactly the same function (returning a map or putting a pin on the map). Or to purchase a commodity on the Web, the user may have similar preferences but the existing e-commerce services provide similar functionalities like searching, comparing, ordering and paying. The user may not necessarily need previous engagement as long as it can provide the required information during the course of the transaction. Generally, determining service relationship, especially their replaceability, needs careful examination.

Our work, unlike other works in the area, considers matching of web Services using Service Names, Outputs and Results as primary factors leaving Inputs for negotiation after matching. We enhance the above matching by: -

1. Computing similarity of service to a request or with each other using service names, results and Outputs. At the same time handling the computation of similarity in cases of multiple outputs and results.
2. The level of matching is quantified. Most approaches use qualitative matching as mentioned above and in all levels of matches, except for direct match, the level or depth of subsumsion is not measured. Our approach quantifies/computes the similarity between two services in consideration of the depth subsumsion and a composite of IOPRs.

The above proposals emanated from our argument that service similarity cannot be judged solely on input and output similarity. Services that return or provide similar or related outputs are not necessarily similar or related by function. Services that have a completely different function may have similar outputs. For instance, if we perform matching based on outputs, we can have exact match for two services returning ISBN number of a book. But in reality, one could be a book searching service and the other could be an automatic ISBN generating service. Similarly, in air

transport WSO one could have services called *getPlaceOfLoading* and *getPlaceOfDeparture*. Both are expected to return code of a place (airport) but the first returns the place where the specific Bill of Lading (air way bill) is loaded while the second one is to get the place where the plane departed from.

A close look at outputs, we noticed that for Services that have real world impact (result) outputs are usually mere textual information or confirmations that notify the result of the operation. For example, for a book selling service that charges the user online and delivers the book at specific address, the output is a notification as the seller successfully charges the required amount from the Customer's bank account and another notification as the delivery is in progress probably with a tracing number of the order. The notifications from the Ontology would look like "*bookOrderIsSuccessful*" and some tracing number for the order. But the impact of the operation to the real world is not expressed here. The results or impacts of the operation like the customer's bank account being deducted and the book being shipped to the mentioned destination are not expressed.

The examples discussed above depict the inefficiency of output to determine service capabilities due to the existence of similar outputs but different functionalities and the mere textual nature of outputs in case of services with real world impact. In general, output is not sufficient enough to determine service functionality. Our approach utilizes the available service description information to assess the similarity of services. The Web Service Ontology provides the missing standardization of service names and impacts and it shows their hierarchical similarity. The founding assumption of semantic Web service is annotation of key parts of service descriptions - service name, inputs, outputs, results and preconditions.

Therefore, our similarity measure uses a composite calculation of Service name, as expressed in the Service Ontology, service outputs and service results. The similarities s1 and s2 are a composition of similarities between service name, output and result as per Equation (1).

$$sim(s1, s2) = sim(o1, o2) \times \alpha + sim(r1, r2) \times \beta + sim(n1, n2) \times \hat{e} \quad (1)$$

where *sim (s1, s2)* is the similarity measure of *service 1* and *service 2*, *sim (o1, o2)* the similarity degree of outputs of service 1 and service 2 respectively, *sim (r1, r2)* the similarity degree of results of service 1 and service 2 respectively, *sim (n1, n2)* the

similarity degree of the names' of service 1 and service 2 respectively, β+α+ê =1 are smoothing function to determine the contribution of output, result and name to the overall similarities of two services. For our case, the service name is the most prominent criterion followed by output and results. As a result, ê =0.4, α=0.35 and β = 0.25. *sim (s1, s2)* is a function with result between 0 and 1 being 1 for strongly similar services and 0 for disjoints.

Services' relatedness is expressed using *subserviceOf, superserviceOf*, *dependsOn*, *composedOf* and *similarTo* attributes in the Ontology. *dependsOn* attribute express dependencies of services and *superserviceOf, subserviceOf* and *similarTo* expresses similarity while *composedOf* expresses both dependency and similarity. Similarly, Results are expressed in sub/super and similar to relationship. Unlike previous works, we measure the similarity of two services primarily using the service names, outputs and results.

Measuring the similarity of service names, outputs or results of two services is the common challenge of measuring semantic similarity of concepts in ontology. Adopting and modifying the concept similarity formula based on shortest path reported by Leacock and Chodotow [33], Equation (2) is used to measure the similarity of two concepts c1 and c2 (c1 and c2 are similar attributes of service 1 and 2).

$$sim(c1, c2) = 1 - \left(\frac{\min\_length(c1,c2)}{2*\max\_depth}\right) \quad (2)$$

where *sim(c1, c2)* is the similarity measure between concept 1 and 2, *min_length(c1, c2)* is the shortest path from concept 1 to 2 following super/sub and IS-A relations for Outputs and Results while *composedOf* is also used in service names, *max_depth* is the maximum depth of the ontology.

Equation (2) measures the similarity of two concepts based on the distance between the two concepts regardless of their position in the ontology tree. Similarity measure for concepts remains the same as long as the distance between the concepts is equal. But, observations depicted that Services at upper layer are general purpose and broad and they may have several sub-services that perform actual functionalities. As the depth in the tree increases, the services become specialized and more similar in function. Services at the lower level of the tree are more concrete and are much similar to each other than those at higher level which are very general.

As a result, we introduce a measure to consider the position of the concepts in the Ontology tree in perspective of the maximum depth of the ontology as give in Equation (3).

$$Pos(c1, c2) = \frac{pos(c1)+pos(c2)}{\max\_depth(c1)+\max\_depth(c2)} \quad (3)$$

where *Pos(c1, c2)* is the position of concept 1 and 2 relative to the maximum depth of the Ontology, *pos(c1)* is position of concept 1, *pos(c2)* is position of concept 2 and *max_depth(c1)* and *max_depth(c2)* are the maximum depth of the branches c1 and c2 are defined on respectively.

Finally, the similarity rate between two concepts is computed as

$$depsim(c1, c2) = sim(c1, c2) \times Pos(c1, c2) \quad (4)$$

where *depsim(c1, c2)* is the depth smoothened similarity of Concept 1 and 2, *sim(c1,c2)* is similarity of concept 1 and 2 as per Equation (2) and Pos(c1,c2) is the computation of Equation (3).

4.6.2 Similarity Extraction Algorithm

In Section 4.6.1 we presented the Equations we proposed to compute similarities of concepts. In this section we present the Algorithm we used to extract the raw data of services. The algorithm takes two services (Service 1 and 2) and computes their semantic similarity as per Algorithm 4.1.

```
INPUT: Web Service Ontology, concept 1(s1), concept 2(s2)
OUTPUT: depsim(s1, s2), n number of nodes
Read Web Service Ontology
Routing table
ShortestPath
Begin
st<-s1

//Generate the node routing table
repeat
 until all nodes attached to st are marked visited
 Start from st and find sn (a node next to it)
     Add sn and edge to the routing table
     If all edges of sn are visited
      Mark sn as visited
     else
      st<-sn
```

```
    Do until (sn equals s2 or Sn is a leaf node or sn is
visited)

If there are no records in routing table
  Return fail
Else
//Find the shortest path
 For records in the routing table
   For all the edges in the record
      If edge is not isSimilarWith or has Alternative then
         pathLengh <- pathlength + 1
   Repeat (for all edges)
    If patLength< ShortestPathLength then
       ShortestPath = current record

    Compute sim(s1,s2) as per Equation (5) and (6)
END
```

Algorithm 4.1: Concept Similarity Extraction Algorithm

4.6.3 Outputs and Results Similarity

A service may have one or more outputs and results. Despite the similarity measure between two comparable concepts remain the same, their multiplicity imposes additional challenge. Noting that a service may have more than one outputs and results and the order of definition of these attributes is not guided, the similarity measure shall accommodate the possible scenario of order and type differences. In the case where Services do not have any inputs or results, Service Name will be the only factor determining similarity. Otherwise, the composition of service name, outputs and results will determine service similarities as per the equation provided in Equation (1).

Table 4.1 presents sample services annotated against the same ontology concept, *bookHotel*. Although all the three services are annotated against the same concept, they have different inputs, outputs and results. As shown in the examples, the difference could be in number and order of definition. As all of the services presented in Table 4.1 are annotated against the same concept *bookHotel*, they are exact matching in terms of name similarity. But it can be seen in the example that they have different outputs and results. Often times, outputs will be defined as a composite attributed single value, which will significantly reduce the challenge. But the case is not always and there are cases where attributes are defined individually or it may be necessary to go passing the composite attribute and evaluate similarity at individual member level as they hold the real power.

Table 4.1: Sample Web Service Profile

No.	Output		Name		Result		Input	
	Original	Annotation	Original	Annotation	Original	Annotation	Original	Annotation
Service 1	roomNumber	roomNumber	reserveYourHotel	bookHotel	roomReserved	hotelRoomBooked	dateFrom	checkInDate
	price	unitPrice			cardBlocked	amountBlocked	dateTo	checkOutDate
	discount	discountAmount			schdulePickUp	shettleBooked	smokingOrNot	roomType
	totalPrice	totalPrice					card_number	creditCardNumber
	vatAmount	totaTax					guestsNumber	numberOfGuests
	serviceCost	serviceFee					withBreakfast	isBreakFastIncluded
	grandTotal	totalPayable					withShuttle	isAirporShuttleRequired
	currency	currencyOfPrice						
	complements	complementaryServices					otherServices	otherIncludedServices
Service 2	cost	unitPrice	bookHotelRoom	bookHotel	roomBooked	hotelRoomBooked	startDate	checkInDate
	-	discountAmount			cardBlocked	amountBlocked	endDate	checkOutDate
	totalCost	totalPrice			notifyShettle	shettleBooked	roomType	roomType
	-	totaTax					cardNumber	creditCardNumber
	-	serviceFee					totalGuests	numberOfGuests
	taxInclusive	totalPayable					bfIncluded	isBreakFastIncluded
	-	complementaryServices					shutleNeeded	isAirporShuttleRequired
	currency	currencyOfPrice						otherIncludedServices
	roomNo	roomNumber						
Service 3	-	roomNumber	setHotelBooking	bookHotel		hotelRoomBooked	startDate	checkInDate
	cost	unitPrice				amountBlocked	endDate	checkOutDate
	-	discountAmount				shettleBooked		roomType
	totalCost	totalPrice			bookingOrder	placeBookingOrder		creditCardNumber
	-	totaTax						numberOfGuests
	-	serviceFee						isBreakFastIncluded
	-	totalPayable						isAirporShuttleRequired
	-	complementaryServices						otherIncluedServices

In the latter two cases, measuring the similarity has two additional challenges.

1. Mismatch: - the number of outputs of one service varies from another one's. There are cases where outputs defined in one service are not totally mentioned in the other one. In the Table 4.1, the case is shown in Service 1 and 2. Service 1 defines 9 outputs while Service 2 defines only 5. Which means that there is a total difference of 4 attributes.
2. Order Difference: - the slightly simple challenge is a difference in the order of attribute definition. It is obvious that the order of outputs/results defined in different services are highly unlikely to be in the same order.

As we noted in the previous sections, outputs are the main measures of service similarity. In this perspective, the first case imposes a main issue as a Service may lack the critical output/s that the entire functionality depends on. In the examples above, Service 3 has not defined any room number as its output which is probably the most critical feature of a hotel booking service. On the other case, Service 1 defines additional outputs which will not affect the main functionality of the service but provide additional information and flexibility while Service 2 defines only the basic outputs of a standard hotel booking service shall have (as per our example).

Despite Service 1 provides more information, it provides the intended purpose of booking a hotel room while there is no way of ensuring Service 3 can do the same. Consequently, for a human intuitive judgement Service 1 and 2 exhibited enough similarity while Service 3 does not.

The example discussed for Service 1 and 2 depicts that a direct comparison of outputs and results has a shortcoming as there is no clue that defines which attribute holds significant information of the service. Accordingly, we suggest a new approach of measuring similarity by comparing the outputs and results of two services against a reference service profile (from the service ontology) as given by Equation (5).

$$sim(or1, or2) = \frac{\sum_{i=1}^{n} match(or1,or2)}{n} X \frac{\sum_{i=1}^{m} match(orOfref,or1)}{m} \quad (5)$$

where *sim (or1, or2)* is the similarity measure of *service 1* and *2's* output/result, *match (or1, or2)* is the degree of match between outputs/result of service 1 and service 2 respectively, *match (orOfref,or2)* is the degree of match between outputs/results of the

reference service and service 1 respectively while *n* and *m* are the number of outputs/results of Service 1 and the reference service.

The final similarity is leveled to between 0 and 1.

$$finSim(or1, or2) = \begin{cases} sim(or1, or2), & sim(or1, or2) \leq 1 \\ 1, & sim(or1, or2) > 1 \end{cases} \quad (6)$$

where *finSim(or1, or2)* is the final similarity measure of *service 1* and *2's* output/result. The result of fin*Sim(or1, or2)* is a number between 0 and 1 where 0 is disjoint and 1 is strong match.

In Table 4.2 we presented three services annotated against same ontology concept having different number of outputs. Assuming that the *bookHotel* services is already defined in our service ontology with 5 outputs, the computation of our equation is demonstrated in Tables 4.2 and 4.3. for the case of simplicity and demonstration, we assumed that the outputs of all services are direct matches, if they exist.

Table 4.2: Output Match Computation for The Services in Table 4.1

No. of Outputs		**9**	**5**	**5**	**2**
	match()	Service 1 (S1)	Reference (Ref)	Service 2 (S2)	Service 3 (S3)
9	Service 1(S1)	9/9=1	9/5=1	9/5=1	9/2=1
5	Reference(Ref)	5/9=0.56	5/5=1	5/5=1	5/2=1
5	Service 2(S2)	5/9=0.56	5/5=1	5/5=1	5/2=1
2	Service 3 (S3)	2/9=0.22	2/5=0.4	2/5=0.4	2/2=1

Table 4.3: Computation of Output Similarity for The Services in Table 4.1

sim()	**Service 1 (S1)**	**Reference (Ref)**	**Service 2 (S2)**	**Service 3 (S3)**
Service 1 (S1)	1.0	1.0	1.0	1.0
Reference (Ref)	0.6	1.0	1.0	1.0
Service 2 (S2)	0.6	1.0	1.0	1.0
Service 3 (S3)	0.1	0.2	0.2	0.4

Although the similarity of two outputs is computed as per Equation (4), we assumed that the outputs of one service directly matches to another's (for the sake of simplicity of the example). The example services in Table 4.1 have 9, 5 and 2 outputs while the service defined by the Ontology developer (The reference service) has 5. Table 4.2 and 4.3 presented the composite similarity of outputs of the example services computed as per Equations (5) and (6). Our computation demonstrated that similarity of services is not transitive as *sim(s1, s2)* and *sim(s2,s1)* yield different results. An exception to the above computation is a service's similarity to itself. It can be seen in table 4.3 that Service 3 has poor similarity to the reference service, the similarity ranking to itself also goes down. To level this issue, we set an overriding policy that a services similarity to itself is always 1.

Noting that service outputs and results will not be defined in same order by different Providers, a service's output/result will be evaluated against all the output/result of the other services to find the pairs yielding maximum similarity. An output/result will only be matched only once to the maximum matching counterpart.

4.6.4 Inputs Similarity

Inputs have been the primary criteria with outputs in previous matchmaking efforts. But, we argued that the main matching criteria should be service name, outputs and results while inputs shall be left for negotiation between the two actors (Consumer and Provider). Inputs are the necessary data elements that the service provider uses to provide the intended purpose.

Our proposal emanated from the fact that service consumers will not know the exact requirements of a particular service beforehand. Services that provide similar or near similar functionality may require different inputs depending on the provider's implementation approach. The example in Table 4.1: Sample Web Service Profile) presents an example that despite all the three service Providers intend to provide a hotel booking service, their inputs are quite different.

As a result, our model does not consider input similarity measure as primary matching criterion but it uses as a supplementary measure to evaluate the replaceability of services. If two services are matched (as computed using service name, output and result) they are matching services waiting negotiation between the Consumer and Provider to determine the exact inputs but if the two services are also matched by inputs in addition to the above three, the two services are

replaceable without the need of negotiation as they provide the same output and require the same inputs.

The similarity measure of inputs will be kept separate and used during negotiation. Inputs also exhibit a similar phenomenon like outputs, as a result the similarity is measured using Equation (5) and (6).

4.7 Matchmaker (Matching Engine)

So far, we presented the supportive components of our matching model. The service ontology and Service Net are constructed to facilitate the matching process which is the main focus of this work. Service matchmaking or discovery is the process of finding a service, out of the available many, that best addresses the service Consumer's request matching functional and non-functional requirements. In both the semantic and syntactic web services, discovery is a critical process. Services are useless unless they are discovered and used by consumers.

Our design requirement clearly stipulated the expectations that need to be addressed by our model. While machine understandability is achieved through usage of ontology language and service ontologies, the requirement of run-time discovery and matching and run time service replacement are expected to be achieved by this component.

Finding potential candidate services out of the available services imposes a real challenge to any consumer especially on the Web. The process requires significant computational resource and time while most of the time the results are not satisfactory or need human intervention to verify documentations.

Automatic matching process happens without the intervention of a human user in which the Broker and Consumer agents interact to find a service provider that can satisfy the user's query. The components constructed in the previous Sections (4.5 and 4.6) are assistant to this process. While the query formulation is not with in our scope, finding the exact match to a query is the core of this work.

In handling the automatic matchmaking, we proposed a two phased matchmaking process with negotiation between the Service provider and Consumer.

1. Candidate Identification: - this is discovery process performed by the Broker. The Broker is the central element of the matchmaking process which holds the service and domain ontology and the service net. Using these utilities, the Broker is in a better position to identify candidate services for a user query. As the name implies, the process may return more than one service and forward it to the Consumer.
2. Negotiation: - after the Consumer evaluates candidate services, a negotiation process will convene. Negotiation will happen with only one matched service at a time, and when it fails a negotiation another negotiation would be initiated with another one.

We present the details of each phase in sections below.

4.7.1 Candidate Identification

This component of the Broker agent is designed to retrieve best suitable candidates from the repository, service-net, for the user's query. As the final matchmaking is left for the Consumer-Provider negotiation, this component is responsible to analyze the user's query, traverse the service repository, identify candidate services, sort according to relevance, analyze as per the reputation and QoS information and finally return the best candidates to the Consumer.

For the purpose of candidate identification, the Consumer agent is expected to provide two basic requirements to the Broker agent. The first one is the potential name/functionality of a service. The second is the potential output and results of the intended service and optionally the inputs. There are broad and sophisticated ways of objective formulation in achieving similar purpose, as it is out of the scope of our work, we will not go into the details of objective formulation which enables consumers to construct their objectives. For the purpose of our work, we assume a simile query formulated with basic service information as mentioned earlier.

In candidate services identification, our model uses the resources constructed in previous sections- the service ontology and Services Net. The service ontology is an abstract definition of services. The Service Net holds the actual implementations of those abstract definitions, services instances. Accordingly, the submitted service query is first evaluated against the service ontology to identify the abstract definition followed by instance matching at Service Net. In both cases, the Broker has calculated and stored the similarity levels of its profiles, at both ontology and instance levels. This is believed to increase the efficiency and accuracy of the Broker.

Our model uses the Algorithms 4.1 and 4.2 in searching candidate services for service query. There are two possible scenarios of search: -

1. Ontology Matching

In this case, the Consumer has not provided any supportive information in identification of the potential service, particularly the Consumer has not annotated the service name but output and results are annotated. In this scenario we perform a blind search to find the potential service profile that can deliver the expected outputs and results. Once the potential service is identified the case will be similar with informed search.

```
INPUT WSO, service request r (output o, result r)
OUTPUT Candidate Service Profile
BEGIN
     Read WSO
     Read t<-matching threshold
     Candidate<- candidate service profile
     Max<-0

      Do for all nodes in WSO
           s<- current node
           compute ss <- sim(s, r) using Equation (5) and (6)
           if ss >= t and ss > Max then
               Candidate<- s
               Max<- ss
       Repeat

  if can Candidate is empty
    return fail
  else
   return Candidate
END
```

Algorithm 4.2: Ontology Matching

We use Algorithm 4.2 to perform blind search to find a service profile from WSO that has maximum similarity with the request.

2. Instance Matching

In contrary to case 1, the Consumer has clearly annotated the name, output and results of the service query or the Broker performed a blind search and identified the matching service from the ontology (using query's outputs and results). In this case, the Broker already has a matching service from the WSO and it is looking for instance service/s that fulfills the request. Therefore, the search

is on the Service Net. The Service Net is repository of service instances that their similarity to each other is already evaluated and maintained.

At this stage, the candidate service profiles are already known. The search for candidate service instances is performed through evaluation of the service query's similarity with the instance service definition in reference to the definition in WSO as per Equation (5) and (6). The Algorithm 4.3. is used for candidate instance identification.

```
INPUT request<- service query, WSO
OUTPUT candidates<-candidate instance services

BEGIN
Read t<-matching threshold

      Find service instances annotated with similar concept from WSO
      as the request
      If(find) then
            For all selected instances
                  Compute ss<- sim(instance, request) as per Equation
                  (5) and (6)
                  if ss >= t then
                        add the instance to candidates
            repeat (while there are more selected instances)
       else
         return fail
      if candidates is not empty and has more than 1 instances
            sort candidates in descending order
  return candidates
END
```

Algorithm 4.3: Instance Matching

Algorithm 4.3 returns list of candidate services sorted in the order of similarity to the query. The Broker submit this candidate service list to the Consumer for identification of required inputs and negotiation.

4.7.2 Negotiation

The Broker identifies candidate services and the Consumer makes matchmaking using its own preferences and identifies the most appealing Provider. In the mentioned processes both the Broker and the Consumer evaluate the suitability of candidate services in terms of functionality, output and results. The actual parameters (input, output and conditions) are unknown or not decided until this point. The negotiator of the consumer corresponds with its counterpart of the Provider to decide these details. The negotiator accesses Consumer's database and policies during negotiation.

The negotiation in our approach uses the controlled approach. Unlike the open approach, in which the Consumer deals with several potential Providers and negotiates on different agendas – price, terms of delivery, quality of service or product and others, in our case the negotiation agenda is pre-determined. Noting that there are different negotiation algorithms and protocols designed for different cases, the negotiation in our work resembles the Web content negotiation scenario implemented in Web browsers.

Following the advent of numerous devices capable of viewing Web contents, from a very small screen handheld device with very limited computation power to big screen and high resolution displays and technologies, Web authors are now forced to provide different contents (images, layouts, and so on) that can be suitably rendered on the user's device. This results in the development of a set of protocols and algorithms of negotiation to efficiently identify the suitable content out of existing variants.

The objective of the negotiation in our model is to determine the IOPRs that the Provider expects from the Consumer. We are primarily focused on obtaining the most appealing service that serves the required functionality by providing only required information. In other words, the objective of the negotiation is to obtain the service that provides the most satisfying Output and Result at minimum Inputs and at acceptable restrictions and pre-conditions.

1. Negotiation Agenda

Obviously interacting agents have several issues to negotiate and agreed up on for smooth cooperation and information exchanging. Assuming that most of the issues are well addressed by other works, the negotiation in our model primarily focuses on the IOPRs. Unlike the traditional service Providers that advertise their services with predetermined IOPRs and have no flexibility for run time negotiation, Service Providers in our model will have the opportunity to provide alternative or variant of services to be determined through negotiation with the Consumer.

Like Web servers that maintain different representations of content, different file formats, different languages or different size, and reply the content that best suites the Client through content negotiation, Service providers would maintain variants of a service that delivers a similar functionality with varied IOPRs.

Therefore, the agendas of the negotiation in our model are to decide the expected inputs from the Consumer side and determine the outputs and results delivered by the Provider and stipulate the restrictions. For simplicity and manageability, we are not considering negotiation on QoS, financial, quality and other commodity quality attributes or negotiations of service delivery protocols, security and other implementation level agendas.

2. Negotiation Actors Interests and Approach

Like the human negotiators, software agent negotiators have different interests. Each side attempts to maximize its benefit and safeguard its interest. Although we initially assume that our agents are cooperative agents, they have different interests to safeguard and shall have corresponding strategies to achieve their mission.

The Provider's agent has the responsibility of delivering high quality service with maximum functionality (like sales agents are expected to sell the most expensive good at maximum quantity), shall provide alternative service when it doesn't have the service the Consumer was looking for (like sales agents attempt to sell walking customer whatever they have, even though the customer has no plan of for it) and demand the maximum information and impose the maximum restrictions. On the other hand, the Consumer's agent attempts to obtain a better match, exact match or a usable service at none or less information requirement and no or less restrictions.

The Consumer acts like a buyer who wants to by the best commodity at no or cheap cost while the Provider is a retailer that wants to sell the expensive commodity or its alternative at maximum price. The negotiation is to mediate these two conflicting interests. The interests are conflicting or somehow overlapping but the agents are cooperative as the one cannot exist without the other.

The strategies at each side are emanated from their interest: -

- The Provider
- The provider always proposes high quality service for first time enquiries unless the request is specific and exact match is found.
- The provider requests all relevant information that is required by high quality service.
- The Consumer
- The Consumer follows a greedy approach to deliver information – attempts to obtain the service by providing only the relevant information.

- The Consumer abides by information restriction and usage policy set by its user.
- The Consumer attempts to obtain an exact match, it attempts other alternatives otherwise.

3. Negotiation Algorithm

Our model uses Algorithm 4.4 to select the 'best fit' service from the identified service variants (candidates). The Broker provides the Consumer the candidate services based on the computation of their similarity to the request. As discussed in Section 4.6.4 inputs are left for negotiation between the Provider and the Consumer.

In our case, the negotiation is the process of identifying the most suitable service out of the candidate services and its variants based on the conformity and availability of required input data.

The algorithm uses elimination approach in which a variant is tested against the condition and eliminated if it does not satisfy. This algorithm is a modification of the httpd Negotiation Algorithm as implemented by Apache [54]. Algorithm 4.4 selects the 'best' variant by a process of elimination. It performs tests in order. Candidates that are not selected at each test are eliminated. After test, if only one candidate remains, it will be select as the best match and proceed to engage with the selected service

```
INPUT
     Candidates<- candidate service from Algorithm 4.3,
     ss<- sim(instance, request) associated with each candidate
     service
OUTPUT Candidate<- a service that best suits the Consumer
   Read Consumer Data
   Read t<- Consumer's threshold of match
BEGIN
     // Eliminate services that has similarity below the t
     For all Candidates
          If ss < t then
               Remove
     Repeat (while there are remaining Candidates)

     //Check if there are surviving candidates
     if number of Candidates = 0 then
       Return fail
     Else continue

     // Remove services that request in existent or an accessible
     // data
     For all candidates
          For all inputs of candidate
```

```
            If the input does not exist or is not accessible
            or has no alternative or cannot be replaced then
                Remove candidate service
                 Continue
        Repeat (while there are remaining inputs)
    Repeat (while there are more candidates)

    //Check if there are surviving candidates
    if number of Candidates = 0 then
      Return fail
    Else continue

    //Eliminate services that does not have acceptable
    //similarity of inputs with the reference service
    For all candidates
        //Compute similarity of inputs with the reference
        //service profile as per Equation (5) and (6)
        Compute inSim<- sim(inCan, inRef)
        If inSim < t then
            Remove Candidate
    Repeat (while there are more candidates)

    //Check if only one candidate is left
    If number of Candidates = 1 then
      Return candidate<- Candidates[0]
    Else if number of Candidates = 0 then
      Return fail
    Else continue

    //Eliminate Services that has high URR in comparison to the
    //reference service
    For all candidates
        Compute urrRate<- total URR/total URR of reference
        If urrRate > threashould URR rate then
            Remove Candidate
    Repeat (while there are more candidates)

    //Check if only one candidate is left
    If number of Candidates = 1 then
      Return candidate<- Candidates[0]
    Else if number of Candidates = 0 then
      Return fail
    Else continue

    //Eliminate services that require restricted (high Usage
    //Restriction Rate (URR) data items but that are not defined
    //in the reference service
```

```
     For all candidates
          For all inputs of the candidate
               If URR of input > threshold set be the Consumer
               and the input is not defined in the reference
               service then
                    Remove Candidate
     Repeat (while there are more candidates)

     //Check if only one candidate is left
     If number of Candidates = 1 then
       Return candidate<- Candidates[0]
     Else if number of Candidates = 0 then
       Return fail
     Else return list of surviving candidates
END
```

Algorithm 4.4: Negotiation Algorithm

4.8 Consumer Data

The Consumer agent needs to know the user's information and preferences to use them during the negotiation process. The Consumer data component is a repository of the user's data and data usage policies.

The Consumer agent is a representative of the Consumer, often a human user, who has large pool of data and preferences. As we noted, the Service matchmaking at the Broker level is computed using the output and results of services leaving the inputs for determination during negotiation. During negotiation the Consumer agent looks into the Consumers pool of data to determine whether the requested data is available or not, including the restrictions. When the data is available and accessible, the agent provides confirmation while rejects the request when it isn't. The Consumer information ranges from the personal details, name, email, phone number, birth day, to other preferences, personal information and historical information.

Dynamic service matchmaking requires the Consumer to provide required data/information dynamically. Even though a number of service Providers provide exactly the same functionalities, depending on their implementation preference and business need, the input requirement varies from Provide to Provider. As a result, the Consumer shall be flexible enough to provide required information dynamically to cope up with the Providers requirement.

To be useful, the Consumer data should be accumulated and stored in a machine understandable format that can be processed by the agents.

4.8.1 Knowledge Representation

The Semantic Web, as the next generation Web aspiring to implement a machine understandable semantic reach Web, comes up with a set of tools and technologies to represent machine processable knowledge. In our case, our sole intention is to learn, accumulate and represent as much knowledge as possible about the Consumer. Providers often request information/inputs to deliver their intended functionality. An online commerce service needs delivery address and payment information among others. Depending on the type of the commodity it could be necessary to know the Consumers color preference, size, height, age, profession and other. The Consumer may also have preferences for instance, the Consumer may have a preference of transport, train over airplane or some taxi operators than others.

To represent such unstructured and machine processible knowledge requires a different thinking apart from the traditional relational or hierarchical data storage and retrieval methods. The semantic Web provides the capability of describing an object using arbitrary attributes and relations using the knowledge representation technologies of RDF and RDFs.

In addition to the knowledge representation technique, it is also important to have an ontology that is capable of describing the Consumer depending on the type of the Consumer. If the Consumer is a human user, the ontology shall provide all the concepts required to fully describe a human being, same to other Consumer types.

Therefore, we use RDF and RDFS to represent Consumer data/information. It enables us to semantically represent our knowledge of the Consumer as well as infer the knowledge flexibly.

Figure 4.11 presents a typical human user is data, described using a domain Ontology. The data items are semantically linked to the Consumer. It can be easily extracted and make inference against this knowledge either manually or automatically.

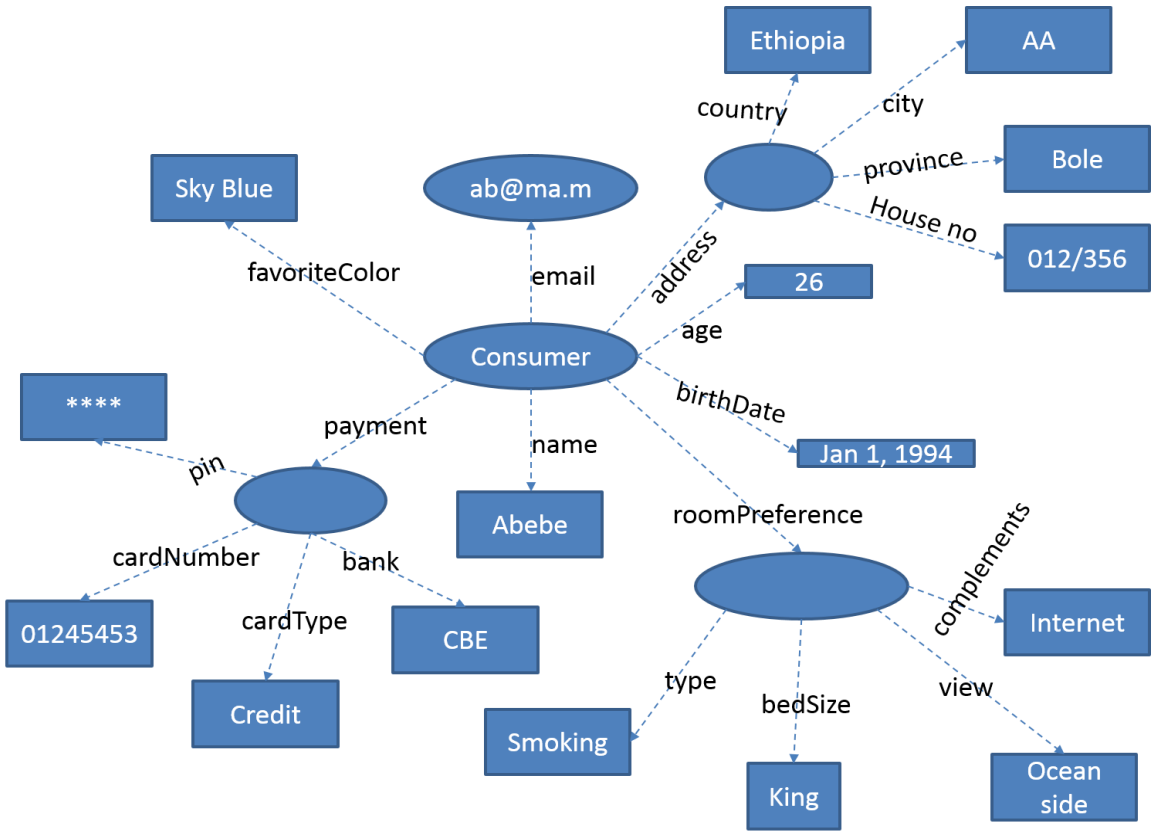

Figure 4.10: Sample Consumer's data

4.8.2 Data Policy

The Consumer information is sensitive and shall be protected from random usage. The user's information security and privacy should not be violated. Because the data is available does not mean that the agent can use it whenever requested. Accordingly, the user should be able to set usage policy specifying how and when to use the information. In setting the policy, it is left for the consumer to determine which information is sensitive or which is not. It is commonsense that a Consumer will not provide his/her payment details without direct authorization.

We define data element usage policy in terms of Usage Restriction Rate (URR). URR is a measure of the data element's restriction level in a number scale from 0 to 10. Where 0 is less restricted or free data element, in which the agent can use whenever it feels necessary, and 10 is highly restricted, which needs the Consumer's direct approval. During negotiation, the agent will try to minimize or avoid the usage of high rated data elements while trying to keep the average data Usage Restriction Rate (URR) to the minimum.

Table 4.4 shows a sample Usage Restriction Rate (URR) assigned to selected data elements. The URR value assigned to data elements is a matter of Consumer's preference. One Consumer may rate email as highly confidential while the other do not.

Table 4.4: Usage Restriction Rate

No.	*Data Element*	*URR*		*No.*	*Data Element*	*URR*
1	email	6		11	roomPreference.bedSize	2
2	address	6		12	roomPreference.view	2
3	address.country	0		13	roomPreference.complements	2
4	address.city	3		14	name	7
5	address.province	5		15	payment	7
6	address.HouseNumber	7		16	payment.Bank	6
7	age	7		17	payment.cardNumber	9
8	birthdate	8		18	payment.pin	10
9	roomPreference	3		19	favoriteColor	3
10	roomPreference.type	3				

4.8.3 Knowledge Accumulation

Knowledge about the Consumer is diverse at the same time services that the Consumer will be interested in need diverse data. For the agent to be useful and act on behalf of the Consumer, it needs to know nearly everything about the Consumer. The agent needs to continuously learn and accumulate the Consumer's information. The agent can learn about the Consumer either explicitly or implicitly. In explicit approach the Consumer itself feeds the relevant information either at one time or at times required while in explicit cases the agent accumulates the Consumer's information from different sources- like the forms the Consumer filled for other purpose or the mail exchange s/he has with her/his medical doctor.

In explicit knowledge accumulation, as the Consumer is engaged, the Consumer can set the URR while it may not be possible in the implicit approach. Accordingly, the agent should apply alternative strategy to learn about the URR of implicit knowledge as well as for the continuous review of URR for both explicitly and implicitly learned knowledge.

4.9 Summary

In this Chapter we discussed the architecture and components of our proposed model. It shows the role and responsibilities of each component and its interaction with other components.

The model is proposed based on the assumptions and requirements provided in Sections 4.2 and 4.3 respectively. The architecture is aimed at dynamic discovery and matching of services. Though the traditional stakeholders are kept intact (the Broker, Provider and Consumer), there is a shift of service selection responsibility from the Broker to the Consumer and while traditional discovery assumes or expects the Consumer to know the parameters of the service, this model leverages the Consumer from knowing or determining the parameters before engagement. The model is believed to provide greater flexibility and enhances automatic service discovery and matching.

This model comes up with new ways of analyzing published services similarity that interconnects into a net of weighted service network. It also introduces new way of computing services similarities with a new concept of service ontology which is a specialized domain ontology that abstractly defines the services of a given domain. Finally, the model introduces a new concept of Consumer data in which the information of a Consumer is stored in a semantic aware environment with a security information attached with it. the Consumer data has a paramount importance for service selection through negotiation.

Chapter 5: Prototype and Results

In Chapter 4 we have presented the architecture, components, algorithms and equations of our model. In this Chapter we will present the prototype implementation, the tools and the environment used to implement our proposed model. We also present the results of experiments.

The implementation of our model requires the development of Service Ontology, Consumer Data and the implementations of the proposed algorithms.

5.1 Prototype

Developing a prototype is one of the expected exercises to experiment the validity and workability of the model. To implement our model, we used a number of existing tools and technologies to develop Ontology, implement the algorithms and make inference as per the given Ontology. In this section we presented the tools we used for the implementation of our model.

5.1.1 Used Tools and Development Environment

Our model heavily depends on semantic Web and knowledge management technologies. The Service and Domain Ontologies require their respective Ontology development and the Consumer Data is a knowledge management phenomenon. Services and Service Queries are also semantically annotated. As a result, we used Ontology and Knowledge development and inference tools to develop the required components of our model.

1. Protégé

Protégé is a free and open-source Ontology editor and framework to develop intelligent systems [55]. It provides a graphical user interface to create, edit and view Ontologies and knowledge bases. It is the leading Ontology development tool that supports Ontology and Knowledge management development, inference, visualization and integration [56]. It has an active community of users and developers and fully supports the latest OWL 2 Web Ontology Language and RDF specification from W3C [55].

Protégé supports different platforms and has Web hosted WebProtégé which enables users to author Ontologies on the Web. For our case we used the Windows based Protégé version 5.0 for the development of the required Ontologies.

1. Apache-Jena

Apache-Jena is an open source Java Framework for the development of Semantic Web and Linked Data application [57]. Jena provides APIs to extract and write data to and from RDF graphs and OWL Ontologies. It provides programmatic environment for RDF, RDFS and OWL and SPARQL [58]. It includes a rule-based inference engine and supports all the variants of Knowledge representation like Turtle, N-Triples and Trig.

Our model uses the Semantic technologies RDF, RDFS and OWL to construct the required Ontologies and the knowledge base of the Consumer (Consumer data). Jena provides the required programing interfaces to programmatically create, update, operate and infer from the existing Ontologies and knowledge. As our preferred implementation language is Java, Jena is the natural match for our work that provides the required utility. We used apache-Jena version 3.7.0.

2. SPARQL Query Language

SPARQL is a set of formalization that provide languages and protocols to query and manipulate RDF graph content [59]. It enables to query and manipulate RDF graph content which resides on the Web or stored locally. It is capable of querying required and optional graph patters along with their conjunctions and disjunctions. It also supports aggregation, subqueries, negation and creating values by expression. SPARQL supports different result formats including XML, JSON, CSV or TSV [57].

It is a W3C recommendation for RDF graph data that has wide acceptance and implemented in Jena. We used SPARQL to make queries and inferences against the Ontologies and knowledge base during matchmaking and negotiation processes.

3. Apache Jena - TDB

Apache Jena – TDB is a Jena integrated Triples storage facility. It provides the APIs to store and retrieve triples created either in RDF, Turtle or other formats [57]. We used TDB1 to store our Ontologies and Knowledge bases.

4. NetBeans IDE 8.0.2

NetBeans is a free and open source Integrated Development Environment (IDE) for Java, JavaScript, HTML5, PHP, C/C++ and others [6]. It has easy integration with frameworks and tools like Jena. We used NetBeans IDE version 8.0.2 for the development of our prototype.

5. Android

Android is a modified Linux kernel-based operating system for mobile devices developed by Google. It is made primarily for touch based handheld devices like smartphones and tablets. It claims to be the worlds most used mobile operating system overpassing its competitors and being used in a variety of devices including watches, Televisions, Vehicles and other electronic apparatuses [7]. We used Android to develop the Consumer component.

6. Android Studio

Android Studio is an Integrated Development Environment for the development of Android Based applications. We used Android Studio version 3.1.2 for the development of the users' agent that runs on android based handheld devices particularly mobile phones and tablets.

5.1.2 Experimental Setting

The prototype's Broker is deployed on a personal computer with 8 GB of RAM, Core i7 Processor and Windows 10 Operating system while the Consumer is deployed on Android 7.0 based mobile phone with 2.1GB RAM.

5.2 Experiment

An experiment is performed to validate the workability of our proposed model in general and the algorithms and equations in particular. The experiment is conducted on a travel domain which includes booking hotels in a given area, finding attractions and other hypothetical scenarios. The experiment includes the development of a domain service ontology, building a service-net and finally a matchmaking for a given query.

For our experiment we used the travel Ontology and Service profiles from the SWASDL test collection and developed an Ontology which attempts to describe the travel industry that enables users to book a flight from origin to destination, book hotel, recommend attractions, restaurants and other events of the Consumer interest. We established 12 hypothetical service providers in the travel industry: 3 hotels, 2 travel agents, 2 restaurants, 3 travel brokers and 2 tourism and event organizers. All the 12 entities published their semantically annotated services to our service Broker. The Providers published their service offerings annotating with the given ontology. As all of the 12 providers work in the same industry, they provide related services with slight differences. We have a total of 150 services offered by all of the 12 Providers.

We created 50 distinct service profiles. This is the Service Ontology that lists all possible service offerings applicable in the industry. Similarly, we capture the information of 5 hypothetical Consumers who used the consumer agent to arrange their traveling. The Consumer agents are fed with the Consumers personal details and preferences.

The Consumers submit their service query using the consumer application installed on their handheld device. The consumer agent submits the query to the Broker where the Broker analyzes its service listing and returns identified candidate services. The Agent makes the negotiation and finally determines the selected service provider. The Consumers submitted a total of 50 service queries against each service profile.

5.2.1 Result of the Experiment

As depicted in Table 5.1, we made a total of 50 Consumer imitated requests. We obtain an average success rate of 77.33% in average time of 30-45 seconds. As it is also presented in Table 5.2, the success rate is higher on average of 84.28% as the number of parameters (inputs, outputs and results) are kept low (less than 10) and significantly reduces to 63.84% as the number of parameters increases.

Table 5.1: Summary Result of Experiment

Number of Queries	Success Rate	Failure rate
50	77.33%	22.67%

Table 5.2: Result of Experiment by Number of Parameters

Number of Inputs, Outputs and Results	Number of Requests	Success Rate	Failure rate
0-10	33	84.28%	15.72%
>10	17	63.84%	36.16%

5.2.2 Comparisons of Equations

We compared our proposed similarity measure equation with other similar notable works. The similarity measure works of Leacock and Chodorow [33] and Wu and Palmer [32] are notable

distance-based similarity measures. We compared our proposed equation with these notable works and the result is presented in Table 5.3.

As it is depicted in the Table 5.3, the results found as per our proposed approach remain consistent to our assessments. As it can be inferred from rows 1 and 4, the distance and depth of service x and y remains the same (50) while the size of the Ontology is increased from 500 to 1000. Our proposed computation results showed consistent result of 0.95 while the others showed fluctuations. On the other side row 2 shows that when x and y are leaf nodes at maximum depth of the ontology, our computation results in 1 (direct match) while Leacock and Chodorow fail to yield a result and Wu and Palmer results in 0.64 which is considerably low similarity.

Similarly, as demonstrated in rows 2 and 3, the distance between nodes x and y remains the same (50) while the depths of the services are high in the Ontology (500/1000). In consistent to our assessment, our computation result showed significant fall in similarity level from (1 to 0.2) while the Wu and Palmer even showed improvement from 0.64 to 0.8.

Table 5.3: Comparison of Semantic Similarity

No.	min_dis (x,y)	pos(x)	pos(y)	max	LCS	max (x)	max (y)	Leacock & Chodorow [33]	Wu & Palmer [32]	Ours
1	50	100	150	500	100	150	100	1.3010	0.8000	0.9500
2	0	500	500	500	323	500	500	-	0.6452	1.0000
3	50	100	150	1000	100	500	500	1.6021	0.8000	0.2438
4	50	950	1000	1000	900	1000	1000	1.6021	0.9231	0.9506
5	186	348	162	8319	67	512	168	1.9514	0.2632	0.7412
6	235	251	486	3610	314	658	622	1.4875	0.8538	0.5566
7	3171	3193	22	7391	2425	6531	28	0.6685	1.5088	0.3850
8	2	486	488	1119	453	1042	886	2.9984	0.9286	0.5050
9	365	1814	2179	9553	771	2243	3838	1.7195	0.3864	0.6441
10	1882	388	2270	7072	25	3848	3046	0.8759	0.0188	0.3342
11	5265	331	5596	2996	1826	1389	9674	0.0561	0.6161	0.0649
12	4525	1049	5575	7995	3879	1220	6174	0.5482	1.1710	0.6423
13	1110	1922	812	4465	628	4380	1452	0.9054	0.4592	0.4104
14	2183	2902	719	5104	908	3569	1870	0.6699	0.5014	0.5233
15	48	60	12	7010	7	884	175	2.4651	0.1819	0.0681

The result in Table 5.3 showed that the findings are aligned with our assessment. As discussed, services at lower level of the Ontology are similar to each other than those at top level. Even

though the distance between services x and y remain the same, as their depth increases, their similarity also increases and vice versa.

5.2.3 Summary

Our model is prototyped and experimented using the available Semantic Web technologies and tools. Developing the full components of the model requires significant effort especially developing the domain ontologies and service ontologies is time taking and challenging. Our prototype focuses on the core areas of the model and experimented on a travel domain with hypothetical entities and users.

The model exhibits good success rates (>84%) in identifying candidate services and making a match when the number of parameters is minimum (less than 10) and are direct matches to the Service Profile. The success rate degrades to 63% when the number of parameters increases and the parameters are not direct matches. It depicts strong character in matching services at different levels.

Similarly, we compared our matching equation with other notable works and we demonstrated that our proposed equation is suitable to our intended purpose yielding results that are consistent to our assessment.

Chapter 6: Conclusion and Future Works

The Semantic Web and Semantic Web Service technologies are efforts towards achieving an objective of transforming the current human understandable Web into machine processable linked data. Web Service matchmaking is an effort in similar objective aiming at enabling software agents and applications to make automated service matchmaking without prearranged effort and human intervention.

Software agents and applications interact with each other without prearranged effort and human intervention and this is an interesting but challenging idea. Matching a service Consumer's functionality request to service Provider's offerings requires beyond the traditional syntax-based approach which is solely based on keyword and direct service signature matching. We worked on a model which aims at incorporating semantic technologies into the traditional Web service technology. In line with the general Semantic Web objective, our objective was to enable Consumer agents to be able to discover and match to a Service Provider automatically and without human intervention.

In doing so, we have reviewed and analyzed up-to-date works in the area, identified the shortcomings and come up with our own proposals. In addressing the objectives of this work, we identified the key architectural components at respective stakeholders (Broker, Consumer and Provider) and proposed working algorithms and equations. Unlike most notable works in the area, our model distributes the burden of matchmaking between the Broker and Consumer agents. The model has a Service Broker which continuously analyzes and measures the similarity among published services and constructs a Service-Net. It also has specialized Service and Domain Ontologies assisting the Service-Net and matchmaking process. The Broker is equipped with our own matchmaking algorithms and semantic similarity measures. The other key stakeholder in our model is the Consumer which is also equipped with a specialized algorithm of candidate service matchmaking and negotiation skill.

Our work introduces new approach of matchmaking that distributes the burden of matchmaking between Broker and Consumer. The main contributions of our work are: -

- Quantified service similarity: - The similarity between two services was not quantified rather was said in qualitative terms. It was often stated in terms of exact, subsume, plug in or disjoint. Our model introduces a formula to compute the similarity level of two services in range of 0 and 1, where 1 is strong match and 0 is disjoint.
- Similarity measure computation equation: - a service's functionality and character are determined by its different attributes. Measuring a service's similarity to another or to a query requires a composite computation of the service attributes. Our model introduces a formula to compute service similarity considering all attributes of services.
- Negotiation: - Service Consumers were expected to know the details (Output, Result and Inputs) of services provided by Publishers. Despite the Consumer may know its expectations for Outputs and Results, it should not be expected for a Consumer to know the input requirements before hand as different implementation of a same service may have different requirements. Our model lifted this requirement from matchmaking process, rather it is determined through negotiation of Service Provider and Consumer.
- Consumer Data: - During negotiations the Consumer has to know and have source data that can be provided to the service Provider. We introduced a semantically annotated information of the Consumer to be semi-automatically organized and used during negotiation process. We also introduced a concept of Usage Restriction Rate which specifies the security of data items in the Consumer data.
- Service Ontology: - the commonly known domain ontologies lack to specify programmatically defined entities and attributes. In other dimension, services have their own attributes that explain the service functionality. In considering the special nature of services, we introduced a specialized service Ontology that specifies services and service attributes.
- Service-Net: - is a directed weighted graph of related services with the weight of the edge indicating the similarity level of the two connected services. It saves the Broker agent from computing matchmakings for every query against all published services while it can easily replay to service replacement requests. It significantly enhances the efficiency of the matchmaking process.

Finally, we conclude our work by pointing out future work directions. This work is a starting point in putting together the newly proposed architectural components, algorithms and Equations. Further improvements are soughed: -

- The contribution of Service Name, Input and Results to the overall similarity of services needs further and careful assessment of large pool of data. Determining the percentage that each attribute (name, output and result) contribute to the overall similarity of services is a main area of future work. Note that, in Equation 4.1 we used our intuitive assessment to determine the dominant factor and its percentage out of Name, Output and Result.
- The success of the matchmaker is highly dependent on the existence of exhaustive, standardized and shared Web Service Ontology. Future works could focus on the possibility of constructing WSO automatically or semi-automatically.
- The interaction between the human user and the Consumer agent is another dimension that needs further work. Enabling the Consumer agent to understand its user's need, formulate it into a service query and then make a request accordingly is a possible future work.
- Automatic binding and consumption of service outputs and results is also another area of work. In the current environment, it is the responsibility of the developer to determine which service to consume and what to do based on the results of the consumed service. To achieve the aspirations of the Semantic Web, the Consumer agent is expected to be general purpose and autonomous, in which it can consume any required service and subsequently act based on the results without expecting further instruction from its human user.
- Security and privacy are the integral parts of any development in the industry. Our work has not considered security requirements except data security requirement defined in form of URR. Securing the entire service selection and interaction environment is another dimension of future work

References

[1] "Web service architecture – W3C Working Draft 2002", the World Wide Web Consortium, retrieved from http://www.w3.org/TR/ws-arch/, Last accessed on May 05, 2015.

[2] S. Ran, "A Model for Web Services Discovery With QoS", Association for Computing Machinery - Special Interest Group on E-Commerce, 2003.

[3] M. Paolucci and S. Carnegie, "Autonomous Semantic Web Services", The Zen of the Web, IEEE Computer Society, 2003.

[4] A. McIlraith and L. Martin, "The Semantic Web - Bringing Semantics to Web Services", IEEE Computer Society, 2003.

[5] G. Priyadharshini, R. Gunasri, and B. Saravana, "A Survey on Semantic Web Service Discovery Methods", International Journal of Computer Applications, Vol. 82 – No. 11, 2013.

[6] M. Burstein and C. Bussler "A Semantic Web Services Architecture -Version 1.0", retrieved from http://www.ai.sri.com/daml/services/swsa/note/swsa-note_v5.html, 2005, Last accessed on May 06, 2018.

[7] A. Adala, N. Tabbane, and S. Tabbane, "A Framework for Automatic Web Service Discovery Based on Semantics and NLP Techniques", Hindawi Publishing Corporation-Advances in Multimedia, 2011.

[8] A. McIlraith, T. Cao Son, and Honglei Zeng, "The Semantic Web - Semantic Web Services", IEEE Intelligent Systems, 2001.

[9] Tim Berners-Lee, James Hendler, and Ora Lassila, "The Semantic Web - A New Form of Web Content That is Meaningful to Computers Will Unleash a Revolution of New Possibilities", Scientific American: Feature Article: The Semantic Web, 2001.

[10] Martin Hepp, "Semantic Web and Semantic Web Services Father and Son or Indivisible Twins?" Peer to Peer, IEEE Computer Society, 2006.

[11] K. Mohebbi, S. Ibrahim, and B. Idris, "Contemporary Semantic Web Service Frameworks: An Overview and Comparisons", International Journal on Web Service Computing (IJWSC), 2012.

[12] C. Cherifi, V. Labatut, and Jean-François Santucci, "Web Services Dependency Network Analysis", Cornell University, 2013.

[13] Ajay D. Kshemkalyani and Mukesh Singhal, "Distributed Computing Principles, Algorithms, and Systems", ISBN 978-0-521-87634-6, Cambridge University Press, 2008.

[14] M. Papazoglou, "Service -Oriented Computing: Concepts, Characteristics and Directions", WISE: 4th international conference on web information systems engineering, pp. 3-12, 2003.

[15] Kuyoro Shade O., Awodele O. Akinde Ronke O. and Okolie Samuel O.," Quality of Service (Qos) Issues in Web Services", IJCSNS International Journal of Computer Science and Network Security, VOL.12 No.1, January 2012

[16] Hongbing Wang, Joshua Z. Huang, Yuzhong Qu and Junyuan Xie, "Web Services: Problems and Future Directions", Web Semantics: Science, Services and Agents on the World Wide Web, pp. 309–320, 2004.

[17] David Martin, Mark Burstein, Drew McDermott, Sheila McIlraith, Massimo Paolucci, Katia Sycara, Deborah L. McGuinness, Evren Sirin and Naveen Srinivasan, “Bringing Semantics to Web Services with OWL-S”, World Wide Web 10.3 (2007): 243-277.

[18] “OWLS: Semantic Markup for Web Services”, retrieved from http://www.w3.org/Submission/OWLS, Last access on Dec 20, 2017

[19] “Semantic Web Services Ontology (SWSO)”, retrieved from http://www.w3.org/Submission/SWSF-SWSO, Last accessed on Dec 20, 2017.

[20] “Semantic Annotations for WSDL and XML Schema”, retrieved from http://www.w3.org/TR/sawsdl, Last accessed on Dec 20, 2017.

[21] M. Klusch and F. Kaufer, “WSMO-MX: A Hybrid Semantic Web Service Matchmaker”, Web Intelligence and Agent Systems: An International Journal 5 (2008)

[22] D. Wei, T. Wang, Ji Wang and Abraham Bernstein, “SAWSDL-iMatcher: A Customizable and Effective Semantic Web Service Matchmaker”, Web Semantics: Science, Services and Agents on the World Wide Web, 9(4):402-417, 2011.

[23] M. Klusch, “Chapter 4: Semantic web service coordination,” in CASCOM: Intelligent Service Coordination in the Semantic Web, 2008.

[24] Matthias Klush, Patrick Kapahnke, Stefan Schulte, Freddy Lecue, and Abraham Bernstein, “Semantic Web Service Search: a Brief Survey”, retrieved from https://doi.org/10.1007/s13218-015-0415-7, Last accessed on Jun, 2018.

[25] M. Paolucci, T. Kawamura, T. Payne and K. Sycara, “Semantic Matching of Web Services Capabilities”, The First International Semantic Web Conference (ISWC), 2002.

[26] M. Klusch, P. Kapahnke and I. Zinnikus, “Adaptive Hybrid Semantic Selection of SAWSDL Services with SAWSDL-MX2”, Semantic Web and Information Systems, 2011.

[27] P. Plebani and B. Pernici, “URBE: Web Service Retrieval Based on Similarity Evaluation”, IEEE Transactions on Knowledge and Data Engineering, 2009.

[28] G. Stavropoulos, S. Andreadis, N. Bassiliades, D. Vrakas and I. Vlahavas, “The Tomaco Hybrid Matching Framework for SAWSDL Semantic Web Services”, IEEE Transactions on Services Computing, 2015.

[29] Bridget McInnes and Ted Pedersen, “Evaluating Measures of Semantic Similarity and Relatedness to Disambiguate Terms in Biomedical Text”, Journal of Biomedical Informatic, Vo. 46, Issue 6, Dec 2013; pp 1116-1124.

[30] L. Christoph, “Measuring Semantic Similarity and Relatedness with Distributional and Knowledge-based Approaches”, DBSJ Journal, Vo. 14, 2016.

[31] R. Rada, H. Mili, E. Bicknell, and M. Blettner, “Development and Application of a Metric on Semantic Nets”, IEEE Transactions on Systems, Man, and Cybernetics, 19(1):17-30, January/February, 1989.

[32] Z. Wu and M. Palmer, “Verbs semantics and lexical selection”, Proceedings of the 32nd Meeting of Association of Computational Linguistics; 1994; pp. 33–138.

[33] C. Leacock and M. Chodorow, “Combining local context and WordNet similarity for word sense identification”, WordNet: An electronic lexical database. 1998;49(2):265–283.

[34] P. Braun, J. Brzostowski, G. Kersten, Jin B. Kim, R. Kowalczyk, S. Strecker and R. Vahidov, "e-Negotiation Systems and Software Agents: Methods, Models, and Applications",2009.

[35] G. A. Miller, "WordNet: A Lexical Database for English," Communications of the ACM, Vol. 38, No. 11, pp. 39–41, 1995.

[36] I. Niles and A. Pease, "Linking Lexicons and Ontologies: Mapping WordNet to the Suggested Upper Merged Ontology," in Proceedings of the International Conference on Information and Knowledge Engineering (IKE '03), pp. 412–416, Las Vegas, Nev, USA, June 2003.

[37] M. Rathore and U. Suman, "QoS Broker Based Architecture for Dynamic Web Service Discovery and Composition", International Journal of u- and e- Service, Science and Technology, pp. 237-252, 2014.

[38] R. Benaboud, R Maamri, and Z. Sahnoun, "Semantic Web Service Discovery Based on Agents and Ontologies", International Journal of Innovation, Management and Technology, 2012.

[39] J. Samper, F. Adell, L. Berg, and J. Martinez, "Improving Semantic Web Service Discovery", Journal of Networks, 2008.

[40] R. Benaboud, R. Maamri and Z Sahnoun, "Agents and OWL-S Based Semantic Web Service Discovery with User Preference Support", International Journal of Web and Semantic Technology (IJWesT), 2013.

[41] S. Pakari, E. Kheirkhah, and M. Jalali, "A Novel Approach: A Hybrid Semantic Matchmaker for Service Discovery in Service Oriented Architecture", International Journal of Network Security and Its Applications, 2014.

[42] G, Fenza, V. Loia, and S. Senatore, "A Hybrid Approach to Semantic Web Services Matchmaking", International Journal of Approximate Reasoning, pp 808–828, 2008.

[43] J. C. Bezdek, "Pattern Recognition and Fuzzy Objective Function Algorithms", Plenum Press, New York, 1981.

[44] G. Heidary, K, Zamanifar, and N. Nematbakhsh, "A Three Phase Semantic Web Matchmaker", International Journal of Smart Home Vol. 4, No. 3, July, 2010.

[45] Z. Chouiref, A. Belkhir, and A. Hadjali, "Enhancing Semantic Web Services Discovery Using Similarity of Contextual Profile", ICIW 2013: The Eighth International Conference on Internet and Web Applications and Services, Italy, Rome, 2013.

[46] A. Belkhirat, A. Belkhir, and A. Bouras, "A New Similarity Measure for the Profiles Management," in the Proceedings of the 13th Inter- national Conference on Computer Modeling and Simulation (UKSIM), United Kingdom, pp. 255-259, 2011.

[47] A. Zohali and K. Zamanifar, "Matching Model for Semantic Web Services Discovery", Journal of Theoretical and Applied Information Technology, 2009

[48] X. Dong, A. Halevy, J. Madhavan, E. Nemes, and J. Zhang, "Similarity Search for Web Services", Proceedings of the 30th VLDB Conference, Canada, 2004.

[49] Z. Gu, J. Li, and B. Xu, "Automatic Service Composition Based on Enhanced Service Dependency Graph", IEEE International Conference on Web Services, 2008.

[50] Abrehet M. and A. Schill, “A Framework for Dependency Based Automatic Service Composition”, Conference: Business Process Management Workshops, BPM 2008 International Workshops, Italy, 2008.

[51] David Martin, M. Paolucci and Matthias Wagner, “Toward Semantic Annotations of Web Services: OWL-S from the SAWSDL Perspective”, The Semantic Web. Lecture Notes in Computer Science, vol 4825. Springer, Berlin, Heidelberg, 2007.

[52] Tomas Vitvar, Jacek Kopecký, Jana Viskova and Dieter Fensel, “WSMO-Lite Annotations for Web Services”, The Semantic Web: Research and Applications. ESWC 2008. Lecture Notes in Computer Science, vol 5021. Springer, Berlin, Heidelberg, 2008.

[53] Marta Sabou, Chris Wroe, Carole Goble and Gilad Mishne, “Learning Domain Ontologies for Web Service Descriptions: An Experiment in Bioinformatics”, WWW 2005, Chiba, Japan, 2005.

[54] Content Negotiation, retrieved from https://httpd.apache.org/docs/2.4/content-negotiation.html, Last accessed on Oct 16, 2016.

[55] https://protege.standford.edu, last accessed date 06-05-2018 3:21AM

[56] A. Musen and the Protégé Team, “The Protégé Project: A Look Back and a Look Forward”, retrieved from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4883684/, Last accessed on Jun 5, 2018.

[57] Apache Jena, retrieved from https://jena.apache.org/, Last accessed on Jun 06, 2018.

[58] Ayesha Ameen, Khaleel Khan and B. Padmaja Rani, “Reasoning in Semantic Web Using Jena”, Computer Engineering and Intelligent Systems, Vol.5, No.4, 2014.

[59] SPARQL 1.1 Query Language, retrieved from https://www.w3.org/TR/sparql11-query/, Last accessed on Jun 06, 2018.

[60] NetBeans IDE, retrieved from https://netbeans.org/, Last accessed on Jun 06, 2018.

I, the undersigned, declare that this thesis is my original work and has not been presented for a degree in any other university, and that all source of materials used for the thesis have been duly acknowledged.

Declared by:

Name: Nigussie Seid Mohammed

Signature: _____

Date: _____

Confirmed by advisor:

Name: Mulugeta Libsie (PhD)

Signature: _____

Date: _____