Chapter 1

1. Introduction and Scope of the Thesis

1.1 Preamble

In this era, traditional web only deals with web of document rather than web of data. Now-a-days, most of the web content is appropriate for human readable rather than machine understandable. Automatically generated web contents are usually presented without their original structural information found in databases. Typical uses of the conventional web involve people’s seeking and making use of information, searching for and getting in touch with other people and viewing academic materials. These activities are not particularly well supported by software tools. Apart from the existence of links that establish connections between documents, the main valuable indispensable tools are search engines. The present document of web deals only with hyperlink HTML (HyperText Markup Language) documents. On the other hand, World Wide Web (WWW) is a global space where anyone can publish their documents for the rest of the world to see. They can hyperlink several documents to any other documents through the server. Most importantly, the web browsing is independent of place and operating system being used in server and clients. The traditional web is made in such a way that its documents only hold the adequate information for the computers to present and publish them, not to understand them. If the documents on the web also contained information that could be used to guide the computers to understand them in the activities like searching, integration of resources and web mining then web mining would be conducted in a much more efficient way. In the era of web 2.0, Google provides the explosion of multi-media by giving significant ranking factor for its mere inclusion, even though indexing bots can hardly discern what it contains. As today’s consumers demand are meet through more accurate and exact results from a search engine query, therefore, web browsing using semantics is the proper tool to satisfy those users demands.
There are lot of keyword based tools available for searching the traditional web like Google, Yahoo, AltaVista, Bing etc. They can only handle web of documents rather than web of data. Moreover, lack of semantics in the present web is a very vital issue. The generated results are extremely perceptive to vocabulary meant i.e. often our initial keywords do not get the results we want. As for example, if we deposit the term “SOAP” (Simple Object Access Protocol) to search engine they produces the outputs like SOAP as an opera, SOAP detergent or SOAP protocol etc. against the said term hence are very much confusing for the consumer to identify the exact one. Clearly, the outputs are only human readable but not the machine understandable. The conventional web does not support to obtain the results in a single web page. As for example if we want to access the information that is spread over various documents in conventional web, we have to provide queries to collect the relevant documents, and then we must manually fetch the partial information and put it together. Overall the existing technology suffers from the limitations in the following areas in terms of knowledge management like searching, extracting, viewing and maintaining information. To alleviate the above mentioned issues, Semantic Web technologies have been applied to the traditional web.

1.2 Brief Overview of the Semantic Web

Semantic web is the extension of the present web. It makes the web enable for automatic retrieval, reuse, resource sharing and integration of information on the traditional web. Semantic Web plays a vital role to access the structured data from the web. In spite of major success of traditional web, it lacks of proper support to web users in terms of searching, extracting and embedding information.

At present, the content of web is not machine accessible due to lack of semantic. So, it is very much essential to include the semantics in the web of resource for meaningful search. Without semantics, computers are unable to understand the meaning of contents of web as well as their structure. Therefore, before entering into the domain of Semantic Web we will demonstrate the definition of semantic web from the view of different scientist and organizations in this field of study.
The following definition has been illustrated by Sir Tim Berners-Lee:

*The Semantic Web is an extension of the current Web in which information is given well-defined meaning, better enabling computers and people to work in cooperation. A web of data that can be processed directly and indirectly by machines.*

-------------------Tim Berners-Lee, James Hendler, Ora Lassila [1]

Sir Tim Berners-Lee originally expressed his vision about the Semantic Web as follows:

"I have a dream for the Web [in which computers] become capable of analyzing all the data on the Web – the content, links, and transactions between people and computers. A ‘Semantic Web’, which should make this possible, has yet to emerge, but when it does, the day-to-day mechanisms of trade, bureaucracy and our daily lives will be handled by machines talking to machines. The ‘intelligent agents’ people have touted for ages will finally materialize.” -------------------Tim Berners-Lee, 1999

World Wide Web Consortium (W3C) has depicted the following vision to improve, extend and standardize the system.

“The Semantic Web is a vision: the idea of having data on the Web defined and linked in such a way that it can be used by machines not just for display purposes, but for automation, integration, and reuse of data across various applications”.

---------------------------------------------W3C Semantic Web Activity Statement [2]

*As a technology, the Semantic Web can be summarized as “knowledge representation meets the Web”. The goal is to create declarative representational notations, i.e. languages, which would enable automatic processing and composition of information in the Web [3].*

The Semantic Web is an extension of the current web that includes documents or portions of documents, describing explicit relationships between things and containing semantic information intended for automated processing by the machines.
1.3 Semantic Web Technologies

The lack of semantic in traditional web has already been discussed in sub-section 1.1. Embedding of engineering and technology method is better than the scientific one to resolve the existing problems in conventional web. At present, the greatest needs are in the areas of integration, standardization, development of tools and adoption by users. Some technologies are necessary for achieving the functionalities, such as Metadata, Ontologies, Logic and Agents.

1.3.1 Metadata

Generally, metadata are data about data. They serve up to index web pages and web sites in the Semantic Web. It also allows other computers to acknowledge what the web page is about. Metadata stores information about a given particular document. More specifically, metadata is a systematic method for describing resources and thereby improving its access. The term refers to any data used to aid the identification, description and location of networked electronic resources. The traditional web have different metadata formats, some of them are quite simple in their description, others quite complex and rich [4].

Sir Tim Berners-Lee [5] introduces a metadata definition in view of Semantic Web: “Metadata is machine understandable information about web resources or other things”. The global use of metadata is pushing the adoption of controlled vocabularies, left aside for decades, as a means to provide shared understanding across different user groups and applications. According to Gill et al [6], new tools and skills are required to implement the use of metadata on the Semantic Web scale. There are different types of metadata and framework exists like Dublin Core, FOAF (Friend of a Friend), vCards, OAI-RE, SIOC and SKOS etc.

1.3.2 Ontology

The term ontology comes from the Greek words ontos (being) + logos (word). In the nineteenth century, it was introduced by German philosophers to distinguish the study of
being as such from the study of various kinds of beings in the natural sciences in Philosophy. Ontology defines the terminology of the domain of interest and provide machine-readable semantics, for enabling enhanced information processing as well as automated information exchange. Ontology is a computational artefact that encodes knowledge about the domain of interest in a machine processable appearance to make it available to information systems.

Ontology gains its momentum in Computer Science and Engineering field during the last few years. It also provides a tool for representing knowledge. Frequently, it is beneficial to have such explicit knowledge about the application domain available for the information system to interact with it at runtime. In a word ontologies allow interactions with additional sharing of explicit domain knowledge.

Ontology is a formal representation of a set of concepts within a domain and the relationships between those concepts. It has been used to understand about the properties of that particular domain. On the other hand, ontology has also been applied to define the domain of interest. Ontologies have been used as a form of knowledge representation about the world. Ontology characterizes the concepts and relationships within a domain. It provides a standardized vocabulary for that domain and the relationships between those concepts. The standardization simplifies a specification which allows a computer to understand the vocabulary.

Ontologies range from simple taxonomies and classifications, to database schemas and on to fully axiomatized theories. Now-a-days, ontologies have been adopted in many big business and scientific communities as a way to share, reuse and process of domain knowledge. Ontologies are now central to many applications such as scientific knowledge portals, information management and integration systems, electronic commerce and semantic web services.
The following definition has been proposed by Sowa in philosophical discipline:

*The subject of Ontology is the study of the categories of things that exist or may exist in some domain. The product of such a study, called an ontology, is a catalogue of the types of things that are assumed to exist in a domain of interest D from the perspective of a person who uses a language L for the purpose of talking about D. The types in the ontology represent the predicates, word senses, or concept and relation types of the language L when used to discuss topics in the domain D.*---------------------------

Sowa 1997.

The following is the definition of Ontology in Semantic Web community as described in [7]:

*An Ontology is a formal explicit specification of a shared conceptualization of a domain of interest.*

If we analyze the above mentioned definition then we can see that it captures several characteristics of ontology as a specification of domain knowledge such as the aspects of formality, explicitness, consensus, conceptuality and domain specificity.

**Formality** – Ontology may be expressed in domain knowledge representation language through formal semantics. These ensure that the specification of domain knowledge in ontology is machine understandable and also interpretable in a well-defined way. The methods of symbolic knowledge representation typically built on the principles of logic which in turn helps to realize this characteristic.

**Explicitness** – To illuminate the knowledge explicitly, ontology plays a vital role after making it accessible for machines. Notions that are not explicitly included in the ontology are not part of the machine-interpretable conceptualization it captures, although human might take them for granted by common sense.

**Consensus** – Ontology reflects an agreement on a domain conceptualization among people in a community. The larger the community, the more difficult it is to come to an
agreement on sharing the same conceptualization. In this sense, the construction of ontology is associated with a social process of reaching consensus.

**Conceptuality** – An Ontology describes a conceptualization in general terms and does not only capture a particular state of affairs. In lieu of making statements in relation to a specific situation involving particular individuals, ontology tries to cover up as many positions as possible that can potentially occur [8].

**Domain Specificity** – The specifications in ontology are limited to knowledge about a particular domain of interest. The narrower the scope of the domain for the ontology, the more an ontology engineer can focus on capturing the details in this domain rather than covering a broad range of related topics.

Ontologies have been adopted in the field of Artificial Intelligence, Computer Science to facilitate knowledge sharing and reuse [9-10]. From the beginning of the nineties, ontologies have become a popular research topic investigated by several Artificial Intelligence research communities including Knowledge Engineering, Natural Language Processing (NLP) and Knowledge Representation (KR). More recently, the notion of ontology has also become widespread in fields such as intelligent information integration, cooperative information systems, information retrieval, electronic commerce, and knowledge management.

The breakthrough behind popularity of the ontologies is in large part due to the following promise: a shared and common understanding of some domain that can be communicated between people and application systems.

According to Guarino et al [11], Ontology is a logical theory which gives an explicit, partial account of a conceptualization. They also describe partial account here means that the formal content of an ontology cannot completely specify the intended meaning of a conceptual element but only approximate it by making unwanted interpretations logical contradictions.
In 1993, Gruber defined the notion of ontology as an “explicit specification of a conceptualization” [12]. On the other hand, Borst defined an ontology as a “formal specification of a shared conceptualization” [13] in the year 1997.

Ontologies play a very important role in information system for mapping, matching as well solving the interoperability among the e-services in any domain of interest. In this proposed work, some domain ontologies have been developed in the field of university e-governance system. There are different types of well-known ontology development tools available in the literature to develop the ontology in any field using Web Ontology Language (OWL). The details of them are given below:

<table>
<thead>
<tr>
<th>Name of the Tools</th>
<th>Organization</th>
<th>Available at</th>
</tr>
</thead>
<tbody>
<tr>
<td>NeOn Toolkit</td>
<td>NeOn Foundation</td>
<td><a href="http://www.neon-toolkit.org">http://www.neon-toolkit.org</a></td>
</tr>
<tr>
<td>Protege</td>
<td>Stanford University</td>
<td><a href="http://protege.stanford.edu">http://protege.stanford.edu</a></td>
</tr>
<tr>
<td>Topbraid Composer</td>
<td>TopQuadrant</td>
<td><a href="http://www.topquadrant.com">http://www.topquadrant.com</a></td>
</tr>
<tr>
<td>Swoop</td>
<td>University of Maryland</td>
<td><a href="http://code.google.com/p/swoop/">http://code.google.com/p/swoop/</a></td>
</tr>
<tr>
<td>Semantic Works</td>
<td>Altova</td>
<td><a href="http://www.altova.com">http://www.altova.com</a></td>
</tr>
</tbody>
</table>

Table 1.1 Ontology Development Environments for OWL

In this proposed work, Protégé Ontology editor has been selected to develop the domain ontology for the sake of the experiment. During the development of domain ontologies, we have adopted some essential features like interrelation, axiomatization and attribution which are common to most ontologies in information systems. Semantic networks have been drawn for generation of the domain of interest to demonstrate the interrelated conceptual nodes. A formal ontology model has been prepared in this regard. Web Service Modelling Language (WSML) has also been applied on the aforesaid model to present the technical constituents of ontology in an accurate way through unifying formal ontology model as proposed by Grimm [14]. Based on the essential characteristics of an ontology introduced earlier, some special types of axioms have been identified during the ontology engineering of proposed work that is common to most ontology languages. The ontology of domain of interest has been drawn which is similar to Unifying Modelling.
Language (UML) class diagrams for technical specifications of information systems. Entity-Relationship (ER) diagrams for the specification of database schemas as both uses interrelation, instantiation. The main purpose of designing software systems through UML class diagrams is to prescribe the technical components of an information system for running on a computer. Once the system runs, the conceptual model behind it has fulfilled its function and is not modified at runtime.

Entity-Relationship (ER) diagrams have been often used only at system design time as an introductory step before designing the relational schema as a basis for efficient storage and data access. They are not intended to be used or changed when the information system is in use. Reasoning with ontologies like verification has been checked during the time of development of axioms in the domain ontology.

Benefits of ontology in the proposed work have been described here briefly. It provides a common and shared understanding regarding certain key concepts in the domain of interest. It caters a way to reuse domain knowledge collectively with ontology description languages such as Resource Description Framework (RDF) schema. RDF schema provides a way to encode the knowledge and semantic so that machines can understand and read. In addition to that it caters automatic large-scale machine processing and unambiguous domain assumptions.

1.3.3 Logic

Logic is the major discipline to study the principles of reasoning. In general, logic offers the formal languages for expressing the knowledge of a particular domain. Logic also provides well-understood formal semantic which reveals that the meaning of sentences. From an algorithmic perspective, implementing logical-reasoning systems demonstrates clearly how complex decidability and complexity are managed [15-16].
**Propositional Logic** - Propositional Logic is a universal logic language providing propositions such as X, Y, Z and logic connectives such as AND and OR. It is decidable.

**First-Order Predicate Logic** - A wealthier meaning to define such propositions by providing terms such as c, f(c, X)........... and predicate symbols that can be applied to these terms P(c), Q(c), f(c, X)............. Terms use the variables that can be existentially or all quantified. It is semi-decidable.

**Second-Order Predicate Logic** - It also provides better logical mechanism as described in [17]. The question of how far one can make progress in simulating second-order features syntactically in a semantic first-order framework has been explored in F-Logic [19] and more generally in HiLog [18].

**Description Logic** - In article [20], authors provide entire family of sub-languages of first-order logic of differing complexity. Common among these languages is to restrict the formalism to unary and binary predicates and to restrict the usage of function symbols and logical connectors to construct complex formulae. The different levels of complexity and the decidability of these languages follow from the precise definition of these restrictions. Therefore, different languages have been defined and implemented, some of them contain intractable worst-case behaviour but however, still work for many practical applications [21].

In the proposed work, the foundation of the semantic web technologies layer has been described. The underlying framework is comprised of a set of design principles and enabling semantic technologies. Some examples of these include Resource Description Framework (RDF), a variety of data exchange formats such as RDF/XML, N3, Turtle, N-Triples, RDF Schema (RDFS) and the Web Ontology Language (OWL). They are supposed to provide a formal description of concepts, terms and relationships within a particular field of knowledge or domain.
1.4 Semantic Web Layer Cake

The Semantic Web has been built not only the basis on mathematical theories but also on fundamental internet technologies. The following Figure 1.1 shows the semantic web layer cake that illustrates the overall Semantic Web architecture: the bottom layers comprised of URI, XML and Namespaces upon which different ontology languages. On top of this layer, languages for proof and trust on the Web are targeted as future work. In the Internet stack, the Internet Protocol components such as IP and TCP are at levels 3 and 4. Sir Tim Berners-Lee introduced a similar conceptual endeavour to structure the Semantic Web in Figure 1.1.

**Figure 1.1 The Semantic Web Layer Cake**

Successful Semantic Web solutions reach out to many diverse sources to perform their larger information need. This fits with the recently formed Web 2.0 philosophy, a
philosophy based on extensive integration and user contributions. Examples include Google maps, Flickr and Facebook etc. Additionally, like any Web 2.0 technology, the Semantic Web benefits with each new additional application and information source.

**Unicode and URI:** Unicode is the standard for computer character representation. On the other hand, URI is the standard for locating resources such as pages on the web. They provide a baseline for representing characters used in most of the languages in the real world. They play a pivotal role for identifying web resources or other things on the web.

**Unicode:** It is an extension of the American Standard Code for Information Interchange (ASCII) that is limited to only 128 characters (a–z, A–Z, 0–9, punctuation). It also possesses a larger set of characters including all letters and signs from all human languages. UTF-8 has been used encoding for Unicode on the web.

The **Uniform Resource Identifier (URI)** is another standard part of the foundational layers of the Semantic Web. URIs have been used to identify unique resources on the web. Depending on their purpose, URIs have been classified in two classes: Uniform Resource Names (URNs) and Uniform Resource Locators (URLs). A Uniform Resource Name (URN) defines an item identity, while a Uniform Resource Locator (URL) provides a method for finding it.

The general syntax to define a URI is:

```
Schema: [/authority][/path][?query][#fragid]
```

Where the schema has been used to distinguish between different types of URI such as http, ftp etc. The authority part is normally identified with a server. On the other hand, the path is identified with a directory or a file on the particular server.

**XML:** XML and its associated standards such as Namespaces and Schemas, form a common means for structuring data on the web but without communicating the meaning of the data. These technologies have already been well studied within the web.
**Resource Description Framework (RDF):** The first layer of the Semantic Web layer cake is Resource Description Framework. It is a simple metadata representation which describes the framework. In general, URI has been used to identify the web-based resources. A graph model has been generated to explain the relationships between web resources over the traditional web. Resource Description Framework (RDF) [22] is a language that uses the XML notation to represent chunks of information as propositions which subsequently form a general graph-like structure. It is a straightforward, powerful data model having the mechanism of describing web resources. RDF can also be represented in the form of Subject-Predicate-Object (SPO) triplets.

The advantage of RDF over XML is that it supports the idea of combining information from multiple sources. The resources and predicates are identified with URIs. An RDF statement itself can be reified by assigning a URI into it. This permits RDF statements to be made of other RDF statements. RDF follows different serialization notations. RDF/XML is a serialization notation to XML, while N3 [23] and N-Triples [24] are more readable formats where each statement is written separately.

**RDF Schema:** a simple modelling language for describing class of resources and properties between them in the indispensable RDF model. It provides a straightforward reasoning framework for inferring the types of resources. RDF Schema [25] can be applied to specify vocabularies (classes and properties) for the purpose of being used in RDF descriptions. RDF Schema makes it possible to illustrate what kinds of properties can be associated to different resources and what are the relationships of those properties.

**SPARQL:** It has been used to query RDF data over the traditional web. The SPARQL has been used to express queries and fetch information from across diverse Semantic Web data sources. The SPARQL query language was developed for the RDF layer of the Semantic Web architecture. In general, SPARQL is a matching graph pattern language that defines a set of graph patterns, the triple pattern. A triple pattern is like a normal RDF triple but with the possibility of a variable instead of an RDF term in the subject, predicate, or object positions.
SPARQL introduced a set of constructs and clauses that could be a part of query. The SELECT clause identifies the variables to appear in the query results and the WHERE clause provides the basic graph pattern to match against the data graph. Additional constructs are FROM that has been used to refer to the name of the graph to be queried. More complex graph patterns can be formed by combining smaller patterns in various ways.

**RIF/SWRL:** It is not defined as a layer on top of RDF in semantic web layer cake. It is therefore somewhat isolated from the other languages associated with the Semantic Web. On the other hand SWRL neatly layered a rule language on top of OWL. Unfortunately, this layering did not capture the essence of either Description Logics.

**Ontologies:** More expressive language for providing more complex constraints on the types of resources and their properties. The details regarding the ontology have already been discussed in section in 1.3.2.

**Logic and Proof:** An automatic reasoning system provided on top of the ontology structure to make new inferences. On the other hand, Proof is no longer a proper layer, that a query language is developed as an alternative to the logic stack.

**Trust:** The top layer of the stack addresses the issues of trust that the Semantic Web can support. This component has not progressed far beyond a vision of allowing people to ask questions of the trustworthiness of the information on the Web, in order to provide an assurance of its quality.

**1.5 Comparative Study between the Traditional Web and the Semantic Web**

The Semantic web and the traditional web are basically two names for the same thing: a global information space where something navigates with URIs. The Semantic Web does exist in the general web and is a part of the traditional web at the same time. This makes them inseparable at the URI-level and consequently, that level is not useful for explaining
the Semantic Web. Basically, the name Semantic Web represents a set of semantically and properly interlinked data units.

Almost, there are two conceptual differences between the Semantic Web and the traditional web:

(1) The Semantic web is a global information space where the information is revealed in a particular machine-targeted language, whereas the traditional web is an information space that contains information targeted at human consumption expressed in a wide range of natural languages.

(2) The Semantic web is a web of formally and semantically interlinked data, whereas the traditional web is a set of informally interlinked information. It is easy to separate links and human oriented textual information. A text document basically means the same thing to the author with or without links. However the use of links can highly increase the understanding and precision of the information to a user.

The semantics of the data are expressed in the machine-targeted language. Thus it is highly depends on how one concept relates to another concept. This is very similar to how humans communicate. Languages used by humans, express meanings by referring to a set of shared concepts that grounds the understanding of their communication. Two applications which are separate from each other have no shared language but may have shared concepts. By creating universal concepts identified by URIs, we can create the set of shared concepts for machines if they need to comprehend the machine-targeted language. The major difference from the traditional web is in the sense that the web is aimed at human consumption through the use of rendering software which ensures that the Semantic web will be used by machines.

1.6 What the Semantic Web is Not

We have already discussed elaborately the traditional web and semantic web including their differences. Now, it is time to alleviate the dilemma over semantic web. The Semantic Web is not Artificial Intelligence. In the particular case of the Semantic Web,
partial solutions are acceptable. Antoniou et al [26] summarize this concept as follows: “If the ultimate goal of Artificial Intelligence is to build an intelligent agent exhibiting human-level intelligence, the goal of the Semantic Web is to assist human users in their day to day online activities”. The Semantic Web is not a separate web but an extension of the current one, in which information are given well-defined meanings, better enabling computers and people to work in cooperation as described by Tim Berners Lee in “The Semantic Web”[27]. The Semantic web neither demands the use of complex expressions nor a rerun of a failed experiment.

1.7 Web Service

Generally, conventional web provides a solution to flexible human-machine interaction whereas web service deal with dynamic machine-machine interaction. Web services connect computers and devices with each other through the Internet to exchange data.

The following definition of web service has been given by the World Wide Web Consortium (W3C) [28]:

“A Web service is a software system identified by a URI [RFC 2396], 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”

According to F. Curbera et al [29], the meaning of web Service is:

“Web services are a new breed of Web application. They are self-contained, self-describing, modular applications that can be published, located, and invoked across the Web. Web services perform functions, which can be anything from simple requests to complicated business processes. Once a Web service is deployed, other applications (and other Web services) can discover and invoke the deployed service.”
1.7.1 Architecture and Technologies of Web Service

Web service is based on the Service Oriented Architecture (SOA) [30-31]. SOA is the latest evaluation of distributed computing which activates software components including application functions, objects and processes from different systems. Based on XML standards, web services can be developed as loosely coupled application components using any programming language, any protocol, or any platform. The architecture of a Web Service can be viewed in two ways firstly, according to individual roles of a web service and secondly, according to its protocol stack.

Normally a web service has the following three common roles in Service Oriented Architecture such as:

i) **Service Provider**: It develops the web service for the service requesters.

ii) **Service Requester**: It invokes a particular web service to fulfil the user’s specific requirements.

Figure 1.2 Web Service Model Recommended by W3C
iii) **Service Registry**: This registry preserves the published web services for further use.

Web Services have been designed to support interoperable machine-to-machine interaction over a network [32]. They have modular applications with interface descriptions that can be published, located and invoked across the Web. It also provides something like Remote Procedure Call (RPC) over the Internet based on existing protocols maintaining widely accepted standards [33].

Web services are characterized by the following three technologies that roughly correspond to HTML, HTTP and URIs in the traditional web architecture [34]:

1. **WSDL** [35] is a normalized way to express service structures operations, messages types, bindings to protocols, and endpoint. Services provide a set of closely related operations associated with a set of resources.

2. **SOAP** (Simple Object Access Protocol) [36] is a message lowest specification that defines a uniform way of passing XML data.

3. **UDDI** (Universal Description Discovery and Integration) [37] provides a repository services that permit the stakeholder to discover available web services.

WSDL, SOAP and UDDI technologies have been discussed in the following sub-section:

**1.7.2 Web Service Description Language (WSDL)**

WSDL is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information.

The operations and messages have been illustrated abstractly. Related concrete endpoints are combined into abstract endpoints. WSDL is extensible to permit description of endpoints and their messages regardless of what message formats have been utilized for communication.
As communication protocols and message formats are normalize in the traditional web, it becomes very important to be able to describe the communications in some structured way. WSDL addresses this issue by defining an XML grammar for describing network services as a collections of communication endpoints so that they can exchange messages.

Port types are abstract collections of operations. A port has been defined by associating a network address with a reusable binding and a collection of ports.

Therefore, Web Service Description Language comprised of the following features:

- **Types**: a repository for data type definitions using some type system.
- **Message**: an abstract type definition of the data being communicated.
- **Operation**: an abstract description of an action supported by the service.
- **Port Type**: an abstract set of operations supported by one or more endpoints.
- **Binding**: a concrete set of rules that handle a specific task and data format specification for a particular port type.
- **Port**: a single endpoint is defined as a combination of a binding and a network address.
- **Service**: a collection of related endpoints.

### 1.7.3 Service Description

This sub-section describes the core elements of the WSDL language and binding extensions for SOAP, HTTP and MIME.

**WSDL Document Structure**

A service is defined as a compilation of network endpoints. There are two levels of service descriptions in WSDL. First, an abstract interface level provides details about types, operations and interfaces of the service. Second, concrete network protocols and
endpoints specify how and where the service can be accessed. The rest of this section describes each of the WSDL elements in detail using the WSDL version 2.0.

Types

The type element encloses data type definitions that are relevant for the exchanged messages. For maximum interoperability and platform neutrality, WSDL prefers to use XSD as the canonical type system.

Messages

Each logical part of Messages is associated with some type of system using a message-typing attribute. The set of message-typing attributes is extensible. WSDL defines several such message-typing attributes to use XSD:

- **Element**: Refers to an XSD element using a QName.
- **Type**: Refers to an XSD simpleType or complexType using a QName.
Port Types

A port type is a named set of abstract operations and messages involved.

The endpoint supports the following four transmission primitives of WSDL:

- **One-way**: The endpoint receives a message.
- **Request-response**: The endpoint receives a message, and sends a correlated message.
- **Solicit-response**: The endpoint sends a message, and receives a correlated message.
- **Notification**: The endpoint sends a message.
1.8 Simple Object Access Protocol (SOAP)

SOAP is document exchange protocol based Remote Procedure Call (RPC) mechanism. It defines a standard protocol specifying how one application communicated and exchange data with another application over the traditional web.

The fundamental approach for expressing data as XML and transporting it across the traditional web using HTTP is simple. In the SOAP protocol, everything that goes across the wire is expressed in terms of HTTP or SMTP header, MIME encoding and a special XML grammar of encoding application data and objects.

A key feature of SOAP and web services is their accessibility. HTTP and SMTP protocols are chosen because they are almost universally available. Put the entire factor together and we have a protocol. SOAP is an XML based protocol used to exchange information throughout the distributed environment.

1.8.1 Anatomy of SOAP Massage

The SOAP specification describes four major components: routing decision in the form of an envelope, protocol binding, encoding rules and RPC mechanism. The envelope defines a convention for describing the content of message, which in turns has implications on how it gets processed.

A protocol binding provides a generic mechanism for sending a SOAP envelope via a lower level protocol such as HTTP. Encoding rules provide a convention for mapping various application data type into an XMLtag based representation.

SOAP document is composed of following:

i) The wrapping of XML inside a SOAP body.

ii) The wrapping of the SOAP body into SOAP envelope.

iii) The optional inclusion of a SOAP header block
iv) Namespace declaration

v) Encoding style directives for the serialization of data.

vi) The binding of the whole thing to a protocol.

SOAP envelope contains two components: a header and a body. Both the header and the body contain multiple block of information.

Figure 1.4 Block Structure of a SOAP Envelope

1.9 Universal Description Discovery and Integration (UDDI): A Repository of Web Services

Universal Description Discovery and Integration (UDDI) is a platform-independent, open industry initiative, eXtensible Mark up Language (XML) based repository. UDDI has been used to register and locate web services. UDDI defines how the services interact
over the web. It provides access to Web Service Description Language (WSDL) documents for describing the protocol bindings and message formats required to interact with the web services listed in its directory.

1.9.1 UDDI Overview

Prior to the UDDI project, no industry-wide approach was available for businesses to reach their consumers and service providers with information about their web services.

Conceptually, a business can register three types of information into a UDDI registry. The specification does not call out these types specifically, but they provide a good summary of what UDDI can store for a business:

**White pages**

It stores the contact information and identifiers about service providers, including name, address, contact information and unique identifiers. This information allows others to discover the web service based upon their identification.

**Yellow pages**

It stores primary information that describes web service using different categorizations such as taxonomies. This information allows others to discover the web service based upon its categorization.

**Green pages**

It stores the technical information that describes the behaviours and supported functions of a web service hosted by business. This information includes pointers to the grouping information of web services and where the web services are located.

1.9.2 How UDDI is used

From a business analyst's perspective, UDDI is similar to an internet search engine for business processes. However, a business exporting a web service needs to expose much more than a simple URL. A business analyst can browse one or more UDDI registries to
view the different businesses that expose web services and the specifications of those services. However, business users probably won't browse a UDDI registry directly, since the information stored within it is not necessarily reader friendly. A series of marketplaces and business search portals could crop up to provide business analysts with a more user-oriented approach to browsing the services and businesses hosted in a UDDI registry. Software developers use the UDDI Programmer's API to publish services and query the registry to discover services matching various criteria. It is conceivable that software will eventually discover a service dynamically and use it without requiring human interaction.

1.9.3 Limitations of UDDI

Even though the Application Programming Interface (API) provided by UDDI allows random searching for businesses, it's not feasible for a program to select new business partners dynamically. Realistically, it's more likely that business analysts with specific knowledge of the problem at hand will use UDDI portals to discover potentially interesting services and partners, and technologists will write programs to use the services from companies that have already been discovered. We'll probably see programs that update the data in a UDDI registry, but most publicly available registries already have a user-friendly interface that allows human users to update information in a registry.

Even though the registries have human-friendly interfaces for direct access, humans should never have to interface with a repository directly. The web service tool we use should automate interaction with a UDDI registry. For example, if we use a tool that creates a web service, that tool should be able to not only deploy the web service into production, but add it to the UDDI registry.

Both business analysts and software developers can publish new business entities and services. Business analysts can use portals attached directly to a particular UDDI server or to a more general search portal that supports UDDI.
1.10 Basic Characteristics of Web Service

Some fundamental characteristics of the web service are as follows:

1. Web services are based on XML messaging which means that the data exchanged between the web service provider and the user are defined in XML. It is self describing via common XML grammar. Messages are exchanged using an underlying protocol such as HTTP or SMTP.

2. Web services provide a cross-platform integration of business applications over the Internet.

3. Since web services are based on loosely coupled application components, each component is exposed as a service with its unique functionality.

4. Web services use standard protocols like HTTP and they can be easily accessible through firewalls.

5. Web services vary in functionality from a simple request to a complex business transaction involving multiple resources.

6. Web services are dynamically located and invoked from public and private registries based on industry standards such as UDDI.

7. A web service offers an interface described by a document that can be processed by a machine in automatic fashion.

1.11 Difference between Web Service and other State-of-the-Art Technologies

1.11.1 Web Service versus Data

Web services are different from data in many significant ways despite similarities in nature and history. First, data in traditional DBMSs are passive objects with a set of known properties, e.g., structure, value, functional dependencies, integrity constraints. It
is just like a source code programme. On the other hand, Web services are active and autonomous entities that have a set of functions rather than values. It is like a source code process. Furthermore, unlike data in DBMSs, Web services may exhibit some run-time performance when they are invoked. Basically, accessing a service on the Web is similar to accessing data from a distributed DBMS. For example, to access a Web service, the service requester must search in one or more service registries. However, Web services carry more complex information, which makes accessing services a more complicated process. This involves understanding the different syntactic and semantic service descriptions, selecting the services providing the requested functionality, understanding their communication protocols and finally engaging in a sequence of message exchange with the selected services.

1.11.2 Web Service versus Web Application

Web service is a programmable unit of software that can be accessed over the web and used remotely. It is used internally by a single application or exposed externally over the Internet for use by multiple applications. Web Services are services with standard interfaces that just expose behaviour. Whereas Web Applications can be defined as all the Web pages such as .aspx, jsp and HTML files, modules, executable code that can be invoked from a Web server. On the other hand Web Applications resides in a web server like IIS, Apache etc. Web Application is a collection of all the web pages. These are the pages which will be displayed when a request is made by the browser. A Web Application is like a web site in which target consumers are people. User interaction is done through web pages, but all data are stored and manipulated on the server.

1.11.3 Web Service versus Application Programming Interface (API)

API and Web service serve as the medium of communication. The only difference is that a Web service facilitates interaction between two machines over a network. An API acts as an interface between two different applications so that they can communicate with each other. An API is a method by which the third-party vendors can write programs that interface easily with other programs. A Web service is designed to provide an interface in a machine processable format usually specified in Web Service Description Language
(WSDL). In general, HyperText Transfer Protocol (HTTP) is the most commonly used protocol for communication. Web service also uses SOAP, REST and XML-RPC as a means of communication.

API has been used any means of communication to initiate interaction between applications. For example, the system calls are invoked using interrupts by the Linux kernel API. An API exactly defines the methods for one software program to interact with the other. When this action involves sending data over a network, Web services come into the picture. An API generally involves calling functions within a software program. The technique used by Web service to expose the API is strictly through a network.

1.11.4 Web Service versus Grid Service

Grid service is basically a web service that conforms to a particular set of conventions. As for example, Grid services are defined in terms of standard WSDL with minor extensions and exploit standard Web service binding technologies such as SOAP and web services security. Therefore, Grid services do look like Web services. In addition to that there is lot of similarities in terms of special forms of distributed computing such as both typically deal with wide-area distributed computing. A Grid service is mainly a Web service with some additions e.g. to make it able to store state information persistently rather than transiently at the server beyond the lifetime of a single request. Web services conceived to share information on the web. On the other hand, Grid Services conceived to share computing power and resources like disk storage databases and software applications. Adding Grid computing to web services is called Grid services. Every Grid service is a Web services, but not every Web service is a Grid services. A WS is usually stateless whether GS is stateful. Grid services extend Web services by stateful services, service instantiation, a two-level naming scheme, a base set of service capabilities, including discovery facilities.
1.11.5 Web Service vs. Remoting

Web services are perfectly suited for consumers outside the firewall calling components on web server. If both consumers and components are inside the firewall, Web services may work in good health. However, all of data travels through a Web server, which can slow the performance. To speed things up, Microsoft provides a binary mechanism called Remoting. Remoting is the .NET technology that replaces DCOM allowing us to communicate between a consumer application and components in a binary format. As a result, remotable components are faster than Web services. Conversely, creating remotable components is more difficult because we must add additional code to components. This code is not much more complicated than its Web service counterpart, but we cannot directly instantiate a remote component. The HTTP channel really uses the SOAP protocol to transfer messages to and from remotable objects.

1.11.6 Web Service versus Service Oriented Architecture (SOA)

Service Oriented Architecture (SOA) is an architectural blueprint where units of work are organized as loosely coupled independent services. In such architecture, business functionalities are provided and consumed as services. Web services are loosely coupled implementations of specific functionalities which can be consumed over the internet as well as intranet. A web service is a function call to an application system. On the other hand SOA is service oriented framework that enables a series of those Web services to occur.

1.11.7 Web Service versus Web Server

According to Wikipedia, a web server is a computer programs that delivers content of the server such as web pages through the Hyper Text Transfer Protocol (HTTP). Web service is an Application Programming Interfaces (API) that can be retrieved over a network, such as the internet and executed on a remote system hosting the requested services. Though both are some sort of a computer program but with different purpose. Web Server's have a specific purpose of delivering web content whereas Web Services have a broader role to play.
A web service executes on a web server. A web server is a portion of software designed to provide web pages, web sites and web services to the end user. IIS, Apache and many more are the typical examples of web server. Web Server is necessarily an HTTP server, a Web Service is simply a service that is available through the network.

1.12 Web Service Protocol Stack

Web service platform is combination of HTTP and XML. Web services use XML language to code and decode data and SOAP for establishing sessions to exchange information over the web.

General web service protocol has the following four layers:

- **Service transport:** This layer is used to transfer messages between several types of applications. It generally supports HyperText Transfer Protocol (HTTP), Simple Mail Transfer Protocol (SMTP) and File Transfer Protocol (FTP) etc.

- **XML messaging:** The messages are encoded in a general XML format to be understandable to both the service requester and the service provider in this layer.

- **Service description:** It describes the interface definition of the web service using the WSDL.

- **Service discovery:** The services are centralized into a common registry generally the Universal Description Discovery and Integration (UDDI) in this layer.
The following Figure 1.5 gives the pictorial view of the web service protocol stack.

![Figure 1.5 The Web Service Technology Stack (Adapted from the W3C Web Service Architecture Document)](image)

1.13 Fundamental Technologies of Web Service

The technological foundations of web service largely rely upon technologies implementing core standards for the World Wide Web. The World Wide Web’s technological foundations are basically those which check the identification of resources Universal Resource Identifier (URI), the representation of their state and the standard protocols that support the interaction between agents and resources in the web space, such as HyperText Transfer Protocol (HTTP).

XML-based standards and technologies include the eXtensible Markup Language itself, XML Schema, SOAP, Web Services Description Language (WSDL), Universal Description Discovery and Integration (UDDI).
The Web service stack categorized the technology of the web service into layer model. The stack starts at the bottom with the basic technologies that allow the data transfer form one machine to another machine. Each layer builds on the lower layer and adds higher-level of abstraction. The upper layer does not depend upon the lower layer and vice versa.

<table>
<thead>
<tr>
<th>Layer Description</th>
<th>Implementation(s)</th>
<th>Other Concern</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard Massaging</td>
<td>Electronic Business XML initiative (ebXML)</td>
<td>Quality of Service</td>
</tr>
<tr>
<td>Service Composition</td>
<td>Business Process Execution Service for Web Service (BPEL4WS)</td>
<td>Management</td>
</tr>
<tr>
<td>Service Registry</td>
<td>Universal Description, Discovery and Integration (UDDI)</td>
<td>Security</td>
</tr>
<tr>
<td>Service Description</td>
<td>Web Service Description Language (WSDL)</td>
<td>Service Development</td>
</tr>
<tr>
<td>Service Massaging</td>
<td>Simple Object Access Protocol (SOAP)/ Extensible Markup Language (XML)</td>
<td></td>
</tr>
<tr>
<td>Service Transport</td>
<td>Hyper Text Transfer Protocol (HTTP), Simple Mail Transfer Protocol (SMTP), File Transfer Protocol (FTP)</td>
<td></td>
</tr>
</tbody>
</table>

**Table 1.2 Web Service Enabling Standards**

**Service Transport:**

The important function of this layer is to transfer data from one machine to another. This layer has been used for the transfer of the data for web services through HTTP protocol.

**Service Massaging:**

This layer has been used to express the format of the data for transferring from one service to another over the transport. Basically, two formats like XML and SOAP have been used.
**Service Description:**

The service description specifies three aspects of the service: i) Operations the service has to made available ii) Messages the service will accept iii) The protocol to which the customer must bind to access the service. Web services use the WSDL to specify a service.

**Service Registry:**

Web service supports the concept of dynamic discovery of e-services. A customer of a service registry finds the services through UDDI. This is a web service repository that supports a standard set of services and allows a web service consumer to dynamically discover and locate the description of the web service.

UDDI supports the two following types of facilities:

i) The service provider uses the UDDI directory to publish information about the web services it supports

ii) The web service consumer sends SOAP-formatted XML massages over the web to the UDDI repository to access a list of web services that matches its criteria.

**Service Composition:**

Service composition is an emerging theme in web service method. Actually it describes the ability for combining web services into a business process to provide better services.

**Service Development**

As Web services decoupled the web service interface from the back-end implementation, there are many ways that a developer can integrate applications.
The following four common development practices have been identified to demonstrate such possible scenarios:

1. **Greenfield Practices**: The developer starts from scratch, creating the web service and the application functionality being exposed as a web service.

2. **Bottom up Approach**: The back-end application functionality which has already been exposed as a web service, the programmer needs to design a suitable interface for the same.

3. **Top down Approach**: Start with an existing web service interface.

4. **Meet in the Middle**: It is a combination of the bottom-up and top-down approach.

1.14 **Benefits of Web Service**

The followings are the benefits of Web Service in Service Oriented Architecture:

**Reusability**: Web service plays a significant role in improving software reusability within organizations. Web service can wrap legacy application, databases and components and expose them as reusable services.

The likelihood of reusability depends on several factors that should be improved by the web services: i) interoperability ii) modularity iii) central registry and iv) reduce compile time dependencies.

**Location Transparency**: A service environment has location transparency because the location is stored in the registry. A client finds and binds to a service and does not care where the service is located. Therefore, the organization has the flexibility to move service to different machine or to move a service to external provider. It is also possible to move code from one platform to another.
**Composition:** Developer assembles applications from a pre-existing catalogue of reusable services. Services do not depend on the applications by which they are composed. As the services are independent from each other, developer can utilize these services in many applications.

**Capability and Availability:** A system is scalable if the overhead requires additional computing power that is less than the benefit provided by the additional power. Since clients only know the service interface and not its implementations therefore changing the implementation to get more scalable and available interface, extra burden.

1.15 Limitations of Web Service

Web Service gives us a new level of satisfaction for sharing the information through traditional web. Information sharing and communication between machine-to-machine, application-to-application, business-to-business, and business-to-customer is no more a problem by the virtue of Web Services. But still it is not much co-operative in heterogeneous and everyday changing environment.

Some of the major limitations of web service are as follows:

i) Web Service technologies illustrate simply syntactical aspects of service.

ii) They provide only some rigid set of services that cannot be adapted for a changing environment.

iii) To adapt it in the changing environment, human interaction is required.

iv) Extra information is added to the service to work in heterogeneous environment.

1.16 Brief Overview of Semantic Web Service (SWS)

Semantic Web Services focus on extending traditional web services such that their meaning is embedded in the syntactical description. Web services deal without semantics.
Adding semantic web technologies to the web services caters the semantics among the e-services. A lot of work, especially in academia, is devoted to this space. The development of Semantic Web Service technologies attempt to enrich the underlying Web Service technologies with capabilities supplied by the Semantic Web technologies.

The promises of dynamic selection and automatic integration of software components have been documented to specify the web services standard that is yet to be experienced. This is partially attributable due to the lack of semantics in the current Web service standards. To address this issue, the Semantic Web community has introduced semantic web service.

By meeting the requirements and capabilities of web services in an unambiguous and machine-interpretable form, semantics make the web service capable of automatic discovery, composition and integration of software components.

Semantic Web Service attempts to increase the usefulness of Web Services by extending them with semantic descriptions. The main areas for extension are interface descriptions, incorporating semantic annotations and the modelling of precise state information of Web Services. The vision of Semantic Web Service [38-39] is to combine two technologies web service and semantic web and enable automatic and dynamic interaction between software systems. Web Service technology allows the description of an interface in a standard way but does not signify what the software system does.

In this proposed approach, we have described service publication, discovery and service selection in a consistent manner to work with Semantic Web Services. In addition to that semantic has been added to the web services in proposed middleware for interoperability among the web services. On the basis of these semantic descriptions, Semantic Web Service technologies seek to automate the task of discovery and selection of services, their execution. Goal based web service discovery and the selection among the e-services have been illustrated in the thesis.
1.16.1 The Role of Semantic in Web Service

Identifying the similarity between services is a difficult task since the terminology used to describe web services and requestor service may not be identical. Besides, the structure and type information present in service descriptions will have to be taken into account. Explicit semantics can play a significant role in resolving terminology ambiguities. Embedding semantic in web service enhances the discovery process of the services. Semantic clears the dilemma over the nature of published web services those are syntactically similar but semantically different and vice versa. The need of semantic in the proposed work governs a specific task like discovery and selection.

Amit Sheth et al. [40-41] introduced the following four main types of semantics for capturing the semantic descriptions:

1. **Data Semantics**: The semantics pertaining to the data used by service
2. **Functional Semantics**: Semantics pertaining to the functionality of the service
3. **Non-functional Semantics**: Semantics related to the non-functional aspects of the service such as quality of service (QoS), security or reliability
4. **Execution Semantics**: Semantics related to exceptional behaviours like runtime errors.

![Figure 1.6 The Nature of Semantic Web Service](image)
The essential trait of SWS is therefore the use of languages with well-defined semantics covering the subset of the mentioned categories that are amenable to automated reasoning. Several languages have been used so far including those from the Semantic Web, for example, Resource Description Framework (RDF(S)) and Web Ontology Language (OWL), SWS-specific languages such as the Web Service Modelling Language (WSML).

1.17 Brief Overview of Semantic Web Service Technologies

Web Service Modelling Language (WSML)

The Web Service Modelling Language has been used to specify the ontologies in a more expressive way. The WSML [42] has also been applied for the description of ontologies, goals, web services, and mediators through the Web Service Modelling Ontology (WSMO). In order to implement the above WSML provides necessary semantics for the Web Service Modelling Ontology. WSML treated as a domain-specific language based on the Meta-Object Facility (MOF) [43]. The Meta-Object Facility (MOF) has been used to define and manipulate a set of interoperable meta-models and their corresponding models. WSML-Flight of general WSML supports logic programming based on F-Logic [44].

Technologies for WSML

Web Service Modelling Toolkit is an Integrated Development Environment (IDE) tool that supports the design of WSML descriptions. On the other hand, the WSML2Reasoner framework is a reasoning framework that enables the end user to register ontologies belonging to the different variants of WSML and query the registered knowledge base.

The Web Service Modelling Toolkit (WSMT)

The Web Service Modelling Toolkit is an IDE for semantic web service that permits developers to develop ontologies, web services, goals, and mediators through the Web Service Modelling Ontology (WSMO) formalism. WSMT also follows the software engineering tasks like SDLC for the Semantic Web Service.
WSML2Reasoner

WSML2Reasoner is a more expressive modular framework that integrates various normalization and transformation algorithms to enable the translation of ontology descriptions in WSML.

Web Service Modelling Ontology (WSMO)

At the outset, WSMO [45-46] was the Web Service Modelling Framework (WSMF) [47]. This framework has been extended through a formal ontology. A formal family of language, the Web Service Modelling Language (WSML) [48] has also been introduced to define the framework. Web Service Modelling Environment (WSMX) [49] environment performed the execution monitoring process.

The following four main features of WSMF have been adopted in WSMO to specify the Semantic Web Services:

(i) **Ontology** that provides the terminology used by other element

(ii) **Goal** that defines the problems of user

(iii) **Web Service** specifications define various aspects of a web service

(iv) **Mediator** which facilitates the resolution of interoperability problem

WSMO is a conceptual framework which creates of semantic descriptions of web services. The actual mediation task has been performed by a specialized component that is being described by a WSMO mediator. WSMO provides a conceptual model for Semantic Web Services that combines the design principles of the web, the Semantic Web.

WSMO [50] provides ontological specifications for the core elements of Semantic Web Services based on the following design principles:

i) **Web Compliance**
ii) Ontology Based

iii) Centrality of Mediation

iv) Ontological Role Separation

v) Description versus Implementation

vi) Execution Semantics

Top level elements of WSMO have been depicted in the following Figure 1.7.

![Figure 1.7 Top Level Notions of WSMO](image)

1.18 Web Service Modelling Execution Environment (WSMX)

There are various types of semantic web service engine exists in real world like Web Ontology Language for Services (OWL-S) [51], Web Service Execution Environment (WSMX) [52], Internet Reasoning Service (IRS) [53], and Managing End-to-End Operations Semantics (METEOR-S) [54] etc. All the aforesaid tools deal with annotation, matching, orchestration, composition as well as discovery. The Web Service Execution Environment (WSMX) is one of the reference implementations of WSMO and
also of Semantic Execution Environments (SEEs) being standardized within OASIS [55]. But in the proposed work, the WSMX framework has been selected for better enhancement of interaction among the users. WSMX is a component based execution environment for the dynamic discovery, selection, mediation, invocation, and inter-operation of Semantic Web Services based on the WSMO specification or set of user’s requirement. It allows Web Services whose semantics have been formally described to be discovered, selected, mediated and invoked to carry out specific client tasks.

**Architecture of Web Service execution Environment (WSMX)**

The WSMX manager controls the operational flow of the system and keeping a full record of the life cycle of each of the various data events created as a user goal. WSMX provides a service interface described in Web Service Description Language (WSDL) to accept service requester goals.

The WSMX Manager regularly scans for new messages. Once a new message representing a requester goal is picked up, it is decomposed, validated and translated into an internal persistent WSMX representation by the Message Parser. The Selector component selects the web Service that provides the best match for the goal based on service requester preferences. The Data Mediator mediates the data provided by the service requester to the ontology defined by the service provider.

The **WSMO Editor** has been selected to generate WSMO descriptions of web services, ontologies, mediators and goals. WSMX provides a WSDL interface to accept these descriptions.

**Compiler** component compiles the services after validating the descriptions and storing them persistently in the Ontology Repository.

The **Matchmaker** then carries out discovery to match the client goal to a capability of Web Services known to WSMX.
The **Selector** component selects the web service that provides the best match for the goal based on service requester preferences.

**Data Mediator** mediates between the data (instance of ontology concepts) provided by the user goal and those expected by the service provider.

**Invoker** makes the actual web service invocation on the selected web service using the mediated data after the mediation data.

WSMX has been defined as a layered architecture depicted in the following figure 1.8

![Figure 1.8 Web Service Execution Environment (WSMX) Architecture (Taken from the OASIS Semantic Execution Environment Technical Committee)](image-url)
The degree of development and refinement for each of these components is quite heterogeneous since the focus was put on the definition of a generic architecture and the different execution scenarios, such as goal achievement and service selection.

The main aspects behind some of the core components have been introduced in [15].

![Figure 1.9 Web Service Modelling Execution (WSMX) Architecture](image)

**Figure 1.9 Web Service Modelling Execution (WSMX) Architecture**

There are the following two types of components in WSMO interface:

i) **Choreography** caters the necessary information to interact with the web service.

ii) **Orchestration** describes how the web service makes use of other web services.

### 1.19 Potential Benefits of Semantic Web Service

The following are the prospective benefits of Semantic Web Service (SWS):

i) Reduced development costs and time

ii) Facilitates the development of flexible and robust systems

iii) Formal Models promote better Software System Maintenance
1.20 Introduction to Middleware

Middleware is the layer that lies between the operating system and the application components. Middleware provides high-level abstractions to support the coordination and interaction of distributed software components for providing the service as per user need. Middleware acts as a major building block for the development of new complex distributed systems. Since Multi-Agent systems are particularly useful for problem domains characterized by highly inherent distribution of resources. It is true that any Multi-Agent system plays a pivotal role in middleware.

Domain-specific middleware for multi-agent systems typically consider agent communication as the prior means for agent coordination. A communication infrastructure usually provides a management system that enables agents to register and locate one another. Coordination infrastructures offer an alternative for direct message exchange allowing agents to interact indirectly via a shared medium. Since interactions necessary for coordination often take place in a concurrent and distributed environment that is unreliable, middleware is a crucial aspect in SOA of Multi-Agent systems. Since these concerns often crosscut the system, vertical integration with domain-specific middleware is an important aspect of the design of any real-world multi-agent system.

This definition focuses on the fact that the middleware is neither part of the application nor part of the operating system. To clarify the concept of middleware, it is useful to distinguish it from the concept of a framework. While a middleware provides services and abstractions at run-time, a framework is a development environment defined by an API and possibly a user interface and a set of tools designed to simplify application development for a domain or set of domains [57].

As such, a framework may include middleware services for use on run-time. Furthermore, not all definitions of middleware exclude systems that are not distributed. For instance, some consider abstractions of the host environment as middleware. Bernstein simply defines middleware as a service that sits between the platform and the applications, and describes what middleware should do [57]. In general, Middleware is
distributed. It can be implemented for several platforms through the applications across domains. Moreover, it ideally supports standard protocols and the standard API.

That middleware must meet requirements across domains is intended to ensure its reusability. However, these days several Middleware systems are actually more or less domain-specific, in that they include services that are only usable in specific domains. Middleware systems that are not domain-specific are typically relatively low-level, for instance OSGi which implements component handling and communication, but not any actual services [58]. The requirement for implementations for several platforms has the same intention.

A comprehensive view of middleware has been illustrated below in Figure 1.10 [59] provided by Schmidt by organizing a middleware system into several layer. The following middleware has been used for classification of other types of middleware too. The host infrastructure layers hides differences between operating systems by providing common abstractions for communication primitives and other resources. The distribution layer provides high-level distribution models allowing location transparent services, allowing communication, method invocation and the like while hiding lower-level issues like addressing and discovery. Common middleware services are services which are domain-independent.

The role of the common services is to handle global considerations like QoS provisioning, resource allocation and persistence. Programmers can access these services instead of relying on lower level distribution abstractions. Domain specific services are tailored for a particular class of systems, and are usually only reusable within the domain as they embody knowledge of this particular domain. For instance, a service that maps coordinates to symbolic locations can be used by most location-aware applications, but not by applications that do not need location. A service that maps coordinates to fields would only be useful in plant production.
A layered view approach of middleware is useful for design, but does not necessarily carry over directly to actual middleware systems. A middleware based on semantics has been developed here to alleviate the interoperability issues as well as their discovery and selection in optimal way as per user need. Various modules with novel algorithm have been embedded into the proposed framework for providing the semantics for machine readability of services. Agent Swarm has also been included for better enhancement of the middleware. Existing semantic framework is also tagged with this framework for the service discovery and selection process as an extra module.

Figure 1.10 Middleware Layers Adapted from Schmidt [59]
1.21 Scope of the Thesis

The following works have been reported in this thesis.

1. To stumble on the deficiency of interoperability, reuse and sharing of e-services within the middleware (Chapter 2).

2. To design and develop a framework of the middleware for the Semantic Web Service to alleviate the above mentioned problems as described in number 1. (Chapter 2).

3. Some algorithms have been developed and evaluated to discover the goal-oriented web services using Multi-Agent (Chapter 3).

4. To select the exact web service as per user’s need in optimal way, an algorithm has been built up in this regard (Chapter 4).

5. Multi-Objective Particle Swarm Optimization (MOPSO) based Ontology Alignment (OA) has been developed for interoperability among the heterogeneous data in the middleware (Chapter 5).

6. This proposed framework has been applied in e-governance domain for the sake of experiment. The experimental results are shown (Chapter 6).

The proposed middleware interacts web applications through web services, where the adopted semantic enables (i) inference paths for driving the user in the discovery and use the services matching user needs (ii) the composition, mediation and execution of web services for accomplishing user’s goal (iii) the exploration of the scenario’s collective views of captured knowledge by deriving additional links and descriptions that are not explicitly given.

Summary of the aforementioned works are provided below on the basis of the consecutive chapter headings.
1.21.1 Design and Development of Middleware for Interoperability among the Heterogeneous E-services.

Semantic web Services in service-oriented architecture has been rapidly gaining acceptance as a means to handle the heterogeneity among the e-services. We have proposed and design the middleware to solve the heterogeneity, web service discovery, and agent based web service selection between the service provider and service requestor. This framework can be applied for any domain of interest.

We applied in the field of e-governance. Design of semantic middleware in the proposed work provides better services to the user.

1.21.2 Goal Oriented Web Service Discovery as per User Need through Multi-Agent

Web service discovery is the major concern in general web based system. At the time of discovery more than 1000 result appears in the front of user as per their demand. With a growing number of web services providing similar functionalities, more emphasis is being placed on how to find the service that best fits the consumer’s goal. In order to find services that best meet their requirements, the service consumers and/or agent swarm need to know both the detailed information for the services and the reliability of this information. The problem, however is that the current UDDI registries do not provide a method for service providers to publish the extra information of their services, and the published extra information of web services is not always trustworthy. We propose a Semantic Agent for web services discovery in Agent Swarm to acquire the fittest web services as per user demand based on their goal template.

1.21.3 A Novel Algorithm for Web Service Selection in Semantic Middleware

After discovering the web services as per user requirement, it is very difficult job to choose the exact one in optimal way. User has to face lot of problems to select the exact service in terms of complexity, time consuming etc. Here, we propose a framework for dynamic web service selection in service-oriented architecture. The framework is broadly
decomposed into following six main components: Graphical User Interface (GUI), Subgoal Generation Module, Syntactical Selection of Web Service, Semantically Selection of Web Service, Matrix Generation, and Selected Set of Web Services. The detailed workflow of the general architecture of the proposed model is illustrated in this section. The sequence diagram of interaction among the components has been depicted here for the clarification purpose. Every module has their unique activities in this architecture to provide the exact web service as per user desire in optimal way.

1.21.4 A Case Study

The achievement of interoperability, integration, selection and discovery of e-services are key challenges for comprehensive one-stop portal in electronic governance due to the heterogeneous data. It is not possible for a service requestor to access e-services via a single entry point even if these services are actually provided by different departments or several agencies due to lack of interoperability, collaborative work, resource sharing, and integration. E-governance is the name given to the use of electronic services means in order to deliver and provide better, more secure reliable and effective facilities of services. The impact of e-service is nothing but providing a better services, which is more trustworthy and more reliable, secure as it offers a vast number of advantages such as: higher quality services, greater engagement with citizens, greater collaboration between agencies, high productivity, and financial benefits for all. The present One-stop portal is suffering from lack of interoperability, resource sharing, operation integration and concept of collaborative work. As a result, a service requestor has to wait for a long time to fulfil particular requirements as per their demand. In addition to that the electronic transaction of information as it is at present is a time consuming and tiring process. On the other hand the data standards of the e-governance are not adequate and comprehensive particularly in the field of e-governance. A user has to face many problems while accessing the particular services as per their requirements.

The following are the most important problems:

i) interoperability between the service requestor goal and offer services

ii) heterogeneous nature of data
iii) exact discovery of services from the lot of existing e-services
iv) Selection of accurate service in optimal way what they desire as per their need.

In view of the above a semantic middleware has been developed to alleviate the interoperability among the e-services.

1.22 Organization of the Thesis

The rest of the thesis is structured as follows. In Chapter 2 the architecture of the proposed framework has been design and development. A goal based web service discovery using the Multi-gent system is illustrated in Chapter 3. Web services selection based on user demand is depicted in Chapter 4 to select the exact service in optimal way. In Chapter 5 aggregated similarity optimization in Ontology Alignment through Multi Objective Particle Swarm Optimization has been illustrated. Chapter 6 provides the case study for the e-governance system. Finally, in Chapter 7, conclusions are drawn along with a discussion on the scope of further research in this direction.
References


[50] Roman, Dumitru, Holger Lausen, Uwe Keller, Jos de Bruijn, Christoph Bussler, John Domingue, Dieter Fensel “Web Service Modelling Ontology (WSMO)." WSMO Final Draft,D2V1.0, April 13, 2005.


